

# 大连理工大学本科毕业设计（论文）

## 基于博弈论的边缘网络虚拟功能调度研究

The Research on Virtual Function Scheduling  
of Edge Network Based on Game Theory

学院（系）： 软件学院

专业： 软件工程（日语强化）

学生姓名： 王威然

学号： 201693071

指导教师： 夏秋粉

评阅教师： 徐子川

完成日期： 2020年6月2日

大连理工大学

Dalian University of Technology

## 原创性声明

本人郑重声明：本人所呈交的毕业设计（论文），是在指导老师的指导下独立进行研究所取得的成果。毕业设计（论文）中凡引用他人已经发表或未发表的成果、数据、观点等，均已明确注明出处。除文中已经注明引用的内容外，不包含任何其他个人或集体已经发表或撰写过的科研成果。对本文的研究成果做出重要贡献的个人和集体，均已在文中以明确方式标明。

本声明的法律责任由本人承担。

作者签名: 王威然

日期: 2020 年 6 月 2 日

## 关于使用授权的声明

本人在指导老师指导下所完成的毕业设计（论文）及相关的资料（包括图纸、试验记录、原始数据、实物照片、图片、录音带、设计手稿等），知识产权归属大连理工大学。本人完全了解大连理工大学有关保存、使用毕业设计（论文）的规定，本人授权大连理工大学可以将本毕业设计（论文）的全部或部分内容编入有关数据库进行检索，可以采用任何复制手段保存和汇编本毕业设计（论文）。如果发表相关成果，一定征得指导教师同意，且第一署名单位为大连理工大学。本人离校后使用毕业设计（论文）或与该论文直接相关的学术论文或成果时，第一署名单位仍然为大连理工大学。

论文作者签名: 王威然

日期: 2020 年 6 月 2 日

指导老师签名: 夏秋粉

日期: 2020 年 6 月 2 日

## 摘要

在传统网络部署中，专有网络硬件的相关费用巨大，例如部署运行，后期维修、更新的费用。网络功能虚拟化是其中一种解决该问题的方法，该技术将网络功能从专有网络硬件中分离出来，以软件的形式在虚拟机中运行和实现，从而可以使电信运营商以低成本高效管理网络服务，同时该技术可创建更灵活的网络服务，以满足更加多样化的需求。但是，该技术的出现也带来了有待学术界和工业界去解决的两个主要问题，虚拟网络功能部署和虚拟网络功能调度。

本文研究后者，提出了基于 Gale-Shapley 算法的“分布式联盟算法”。有别于 Gale-Shapley 算法用于解决博弈论中经典的稳定婚姻问题（一对匹配问题），本文设计的算法可以解决一对多的匹配问题。在该算法中，虚拟机结点扮演着稳定婚姻算法中“女生”的角色，虚拟网络功能实例扮演着“男生”的角色，通过虚拟网络功能实例请求虚拟机结点并利用延迟接受的思想，最后可以得出一个稳定的联盟结构。本文也证明了纳什均衡，即非合作博弈均衡的存在。

本文利用 Python 进行仿真实验并与现有的“基因遗传算法”与“轮训调度算法”相比，其结果表明“分布式联盟算法”具有更好的调度完成时间，并且虚拟机与虚拟网络功能对于彼此的组合相互满意，具有较优的匹配稳定性。

**关键词：**网络功能虚拟化；虚拟网络功能调度；虚拟网络功能服务链；博弈论

## **Virtual Function Management of Edge Network Based on Game Theory**

### Abstract

In the traditional network deployment, the costs associated with proprietary network hardware are quite huge, e.g. the cost of deployment and operation, as well as the cost of post-maintenance and updating. Network Function Virtualization (NFV) is one of the emerging technologies to solve this problem, which separates network functions from proprietary network hardware, and runs and implements them in virtual machines in the form of software, thereby enabling telecommunications operators to efficiently manage network services at low cost. Simultaneously, NFV can create more flexible network services to meet more diverse needs. However, this technology also brings two major problems to be solved by academia and industry, i.e. virtual network function deployment and virtual network function scheduling.

Our work focused on the latter one and we proposed the Decentralized Coalition Algorithm based on Gale-Shapley algorithm. But different from the Gale-Shapley algorithm solving the Stable Marriage Problem (one-to-one matching problem, a classical Game Theory problem), this algorithm can solve one-to-many matching problem. In this paper, the virtual machine plays the role of "girl" in the Stable Marriage Problem, and the VNF instance plays the role of "boy". Through the VNF instance's requesting the virtual machine node, and the idea of using the deferred-acceptance algorithm, a stable coalition structure (without blocking pair) can be constructed and the existence of Nash Equilibrium (Non-cooperative Equilibrium) can be proven.

The simulation is run via Python. Compared with Genetic Algorithm and Round-Robin Scheduling Algorithm, our algorithm shows better scheduling completion time. Due to the inheritance of the stability of the Gale-Shapley algorithm, the virtual machines and VNF instances are satisfied with their coalitions, which shows a better matching performance.

**Key Words:** Network Function Virtualization; Virtualized Network Function Scheduling;  
Virtualized Network Function Service Chain; Game Theory

## 目 录

摘要 .....	I
Abstract.....	II
1 绪论 .....	1
1.1 边缘计算概述 .....	2
1.1.1 软件定义网络概述 .....	3
1.1.2 网络功能虚拟化概述 .....	3
1.1.3 软件定义网络与网络功能虚拟化的异同 .....	4
1.2 博弈论概述 .....	5
1.3 网络功能虚拟化的理论意义及应用价值 .....	5
1.4 网络功能虚拟化的发展挑战与本文创新点 .....	6
1.5 本文的组织结构 .....	7
2 网络功能虚拟化文献综述 .....	8
2.1 虚拟网络功能的部署与动态网络功能链 .....	8
2.2 虚拟网络功能服务链的调度 .....	12
2.3 本章小结 .....	14
3 基于博弈论的虚拟网络功能管理的模型建立 .....	15
3.1 系统模型 .....	15
3.2 虚拟网络功能服务链 .....	16
3.3 问题定义 .....	16
3.4 本章小结 .....	20
4 分布式联盟构建机制 .....	21
4.1 虚拟网络功能实例与虚拟机的偏好列表 .....	21
4.1.1 虚拟网络功能实例的偏好列表 .....	21
4.1.2 虚拟机的偏好列表 .....	21
4.1.3 阻塞组合与稳定联盟 .....	21
4.2 分布式联盟算法 .....	23
4.3 分布式联盟算法中的纳什均衡 .....	25
4.4 本章小结 .....	26
5 实验结果及其数值分析 .....	27
5.1 实验环境 .....	27
5.1.1 NetworkX .....	27

5.1.2 Matplotlib .....	27
5.1.3 xlrd.....	27
5.2 模型实现细节 .....	27
5.3 测试结果 .....	30
5.4 分析与讨论 .....	33
5.5 本章小结 .....	34
结    论 .....	35
参    考    文    献 .....	36
修改记录 .....	41
致    谢 .....	43

## 1 绪论

在过去几十年的发展中，对于支持无线网络的移动设备的使用快速增长。作为结果，移动数据流量呈指数级增长，并且这种趋势在未来几年仍然会增加<sup>[1]</sup>。同时，众所周知，在所有的无线访问技术中，基于无线上网（Wi-Fi）的技术是使用最广泛的技术，已经成为人们生活中不可或缺的一部分。例如，英国通信局最近报告称，英国 81% 的移动用户经常使用 Wi-Fi<sup>[2]</sup>。尽管具有便利性和成本效益，Wi-Fi 通常以尽力而为的方式工作，因此几乎无法提供诸如访问带宽，端到端延迟和服务可用性等指标的任何服务质量（Quality of Service, QoS）保证。但是，越来越多的新应用将需要此类服务质量保证<sup>[3-5]</sup>，因此非常需要端到端解决方案。这不仅给 Wi-Fi 带来了新挑战，也给互连 Wi-Fi 接入点的有线网络带来了新的挑战。

可以通过利用分片即服务（slice-as-a-service, SaaS）的概念，即应用程序驱动的端到端（end-to end, E2E）网络分片，来解决上述挑战。具体来说，基础架构供应商创建各种网络切片，即虚拟网络，确保切片中的某些类型的服务质量，并将其租给承租人以向最终用户提供不同的服务<sup>[6]</sup>。图 1.1 以未来的 5G 网络为例，在同一基板网络上创建三个切片，以提供高吞吐量，低延迟和实时，低速率和非关键性分别为智能手机，自动驾驶汽车和大规模物联网（Internet of Things, IoT）提供服务<sup>[7]</sup>。

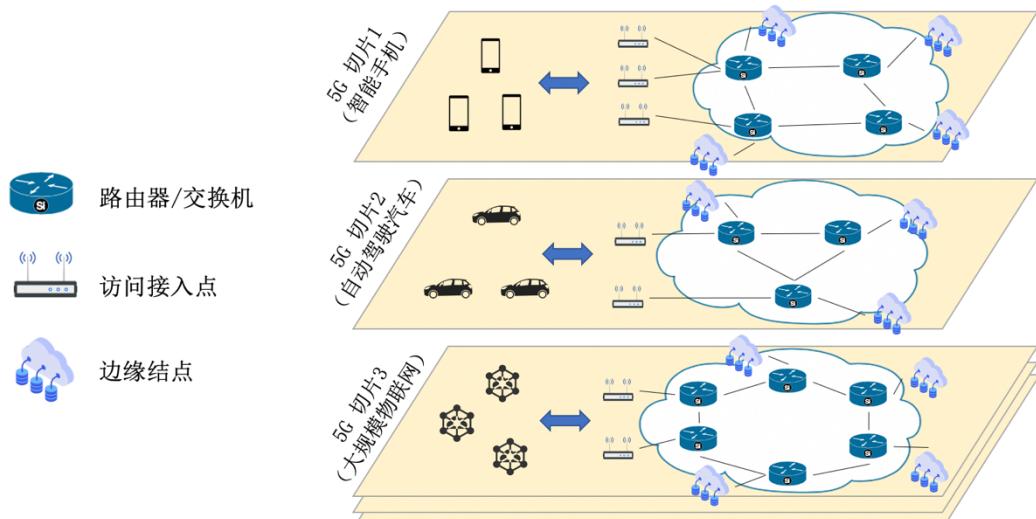


图 1.1 一个 5G 网络中 E2E 应用驱动的网络切片示例

从系统实施的角度来看，软件定义网络（SDN）和网络虚拟化被视为 SaaS 的关键支持技术。同时要确保某些新兴应用的服务质量，基础架构供应商不仅需要网络方面的支持，而且还必须利用基于信息技术资源虚拟化的网络功能虚拟化（NFV）。本章将介绍该虚拟化技术。

## 1.1 边缘计算概述

物联网的激增和云服务的成功推动了一种新的计算范例，边缘计算的发展。它要求在网络边缘处理数据。边缘计算具有解决响应时间要求，电池寿命限制，节省带宽成本以及数据安全性和隐私性的潜力。数据在网络边缘越来越多地产生，因此，在网络边缘也处理数据将更加有效。诸如微数据中心<sup>[8], [9]</sup>，cloudlet<sup>[10]</sup>和雾计算<sup>[11]</sup>之类的先前工作已引入社区。

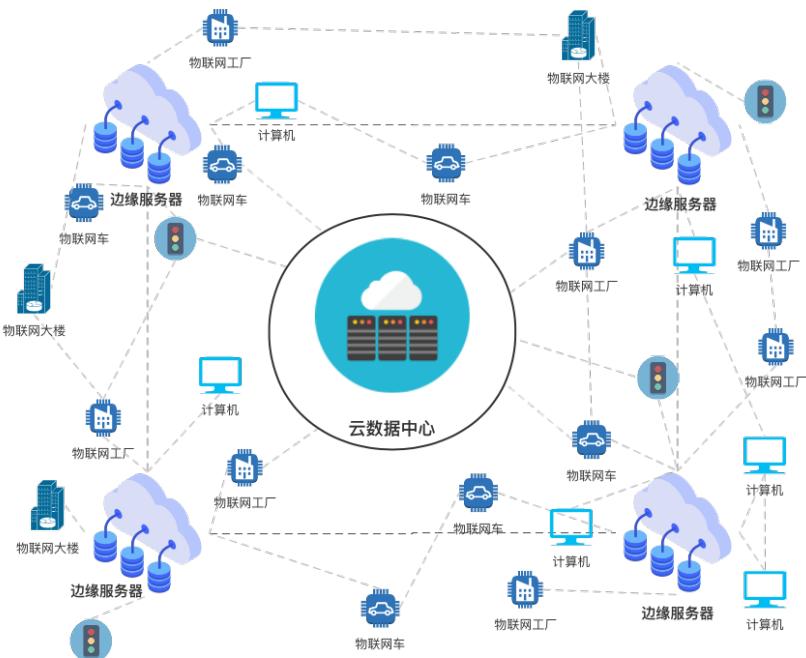


图 1.2 中心云与边缘云计算实例

由于少量数据中心中的集中计算，存储和联网，以及边缘设备和远程数据中心之间的相对较长距离，传统的云基础架构将面临一系列困难。为了应对这一挑战，边缘云和边缘计算似乎是一种有前途的可能性，它可以提供更接近资源贫乏的边缘物联网设备的资源，并有可能培育出新的物联网创新生态系统。包括网络功能虚拟化和软件定义网络在内的一系列新兴技术使这种前景得以实现。

### 1.1.1 软件定义网络概述

2006年，以斯坦福大学教授 Nike Mckewn 为首的团队提出了OpenFlow的概念，并基于OpenFlow技术实现网络的可编程能力，是网络像软件一样灵活编程，软件定义网络技术应运而生。该技术可通过直接编程将网络控制与数据转发脱钩<sup>[12]-[15]</sup>。软件定义网络旨在将应用软件参与网络的控制管理，满足上层业务需求，通过自动化业务部署简化网络运维，消除了很多国际互联网工程任务组(The Internet Engineering Task Force, IETF)的协议。这意味着学习成本的下降，运行维护成本下降，业务部署快速提升。

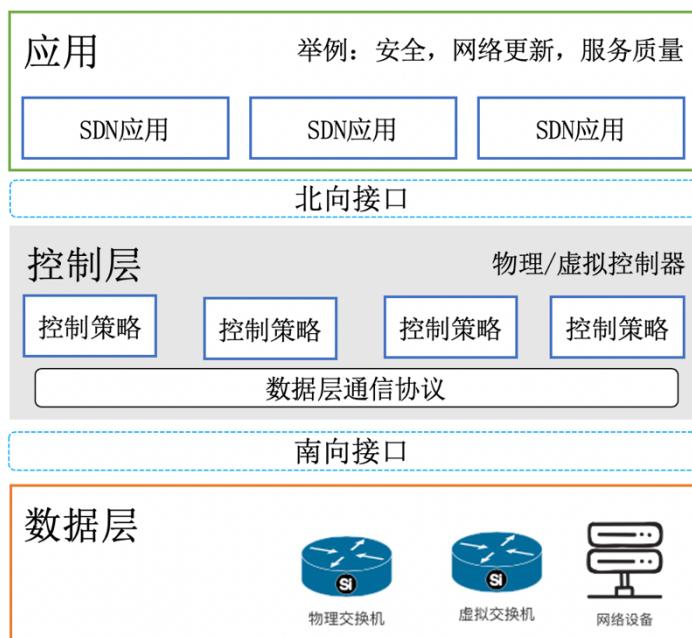


图 1.3 软件定义网络基础架构

### 1.1.2 网络功能虚拟化概述

网络功能虚拟化技术源自如何在高容量多媒体时代改善网络运维这一问题，被大量网络管理人员和通信公司所关注。该技术将网络节点阶层的功能，分割成几个功能区块，分别以软件方式实作，不再拘限于硬件架构。网络功能虚拟化技术用于降低来自众多手持设备的流量空前增长导致服务提供商支持这种过多的网络通信的资产开支(CapEx)和运维开支(OpEx)。同时该技术有望通过将网络服务和底层硬件解耦来提供服务供应的灵活性，以至于该技术可以在商用现货(Commercial Off-The-Shelf, COTS)硬件上运行被称之为虚拟网络功能(Virtual Network Functions, VNFs)的软件映像。尤其是网络功能虚拟化利用虚拟化技术，允许在按需安装过程之后将网络功能部署在任何地方。因

此，服务提供商现在可以提供满足终端用户需求的高度专业化的网络服务，而不必增加用于购买和安装专有硬件设备和中间盒的资本投入。作为一种新兴技术，网络功能虚拟化同时也带了一些挑战。最常见的包括相比于高度专业化和专用的硬件，网络服务的性能以及他们的合理且高效的放置。这些网络功能的放置对于各个虚拟网络功能的性能是个主要的关注点，并且引起了网络功能虚拟化社区的高度关注（例如 IETF），使其成为一个值得研究的领域。关于网络功能虚拟化的挑战及应用价值将在本章后半段详细介绍。

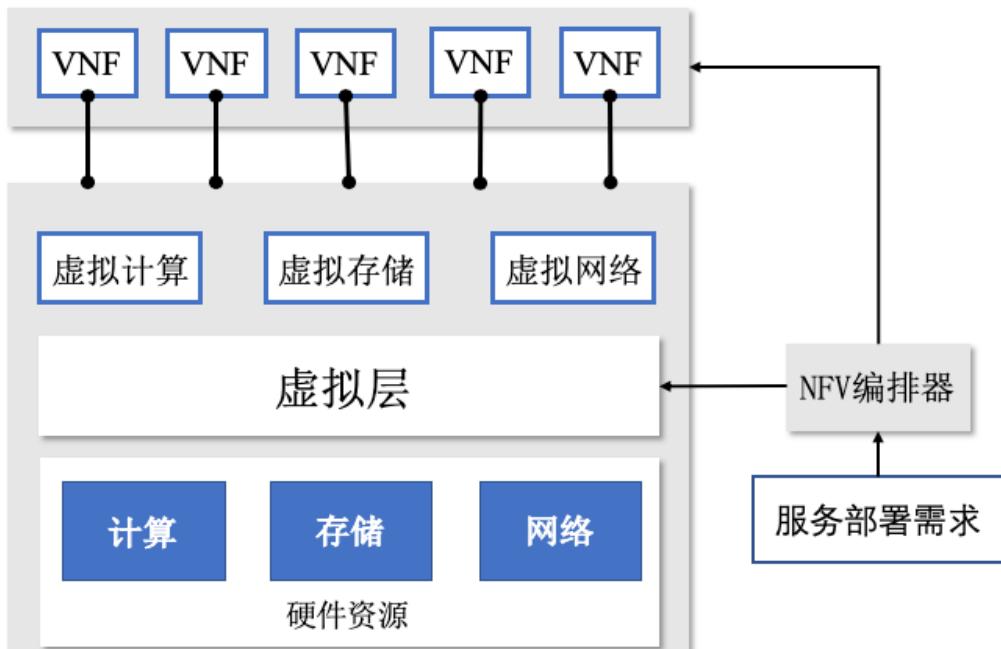


图 1.4 网络功能虚拟化的网络架构

### 1.1.3 软件定义网络与网络功能虚拟化的异同

虽然网络功能虚拟化技术的兴起和各项工作都比软件定义网络起步要晚，但是网络功能虚拟化的更为单一概念在现代网络中扮演者与软件定义网络同样重要的作用，且与软件定义网络互惠互利，彼此高度互补。网络功能虚拟化可以通过虚拟化软件定义网络控制器（可以视为网络功能）在云上运行来为软件定义网络服务，从而允许将控制器动态迁移到最佳位置。反过来，软件定义网络通过在虚拟网络功能之间提供可编程的网络连接来为网络功能虚拟化服务，以实现优化的流量工程和控制<sup>[16], [17]</sup>。但是，网络功能虚拟化和软件定义网络与概念，系统体系结构和功能完全不同，可以从以下几个方面进行概述：

(1) 网络功能虚拟化是一种以软件方式实现网络功能的概念。而软件定义网络是一种实现集中控制和可编程网络体系结构以提供更好的连接性的概念。

(2) 网络功能虚拟化旨在减少资本支出，运营支出以及空间和功耗。而软件定义网络旨在提供网络抽象以实现灵活的网络控制，配置和快速创新。

(3) 网络功能虚拟化将网络功能与专有硬件分离，以实现敏捷的配置和部署。而软件定义网络将网络控制平面与数据平面转发分离，从而通过启用可编程性来提供集中式控制器。

## 1.2 博弈论概述

早在 2500 年前，春秋时期著名军事家孙武所著《孙子兵法》中就含有博弈论 (Game Theory) 思想与雏形，博弈论英文中 game 本意为游戏，常为赌博与棋牌等一系列具有规则的输赢活动，博弈/游戏的参与者同时考虑自身和其他参与者的决策从而做出相应的决策。1928 年“现代计算机之父”冯·诺依曼证明了博弈论的基本原理，随后又将二人博弈推广为多人博弈，奠定了该领域的理论基础。1950 年，约翰·纳什在其博士论文中提出了非合作博弈 (Non-cooperative Games)<sup>[18]</sup>，利用不动点定理证明了均衡点的存在，即现在的纳什均衡 (Nash Equilibrium)。纳什均衡也为本文算法解决边缘网络虚拟功能管理的主要思想及算法设计的主要思路。

现在博弈论应用于计算机科学的各个领域。斯坦福大学教授 Yoav Shoham 在 Computer Science and Game Theory 一文中提出了博弈论是人工智能，计算机图形学理论，电子商务，计算机网络等计算机科学的其他领域的组成部分<sup>[19]</sup>。例如，Hu 和 Wellman<sup>[20]</sup>在基于一般和 (general sum) 博弈对策的多智能强化学习中证明了算法收敛域纳什均衡点。Manshael, Zhu 和 Alpacn 等人<sup>[21]</sup>提供了使用博弈论方法的计算机和通信网络中安全性和隐私研究的结构化和全面概述，以突出博弈论在解决计算机网络和移动应用程序中不同形式的安全性和隐私问题方面的作用。Wang, Sun, Ji 等人<sup>[22]</sup>利用基于博弈论的优化策略迭代更新像素和超像素的概率。在具有挑战性的数据集上进行的实验表明，与现有技术相比，该方法可以获得更好的分割结果。Martinez 和 Quijano<sup>[23]</sup>使用基于博弈论的多代理系统框架来解决工程系统控制中的问题。博弈论在处理边缘网络虚拟功能管理中相关应用将在本文文献综述部分阐述。

## 1.3 网络功能虚拟化的理论意义及应用价值

在传统网络中，所有设备都被部署在专有封闭的平台上，所有网络但具有密封性，从而导致这些硬件都无法共享。网络功能虚拟化与传统网络设计与服务提供截然不同，它将网络地址转换 (Network Address Translation, NAT)，防火墙 (Firewall)，入侵检

测系统（Intrusion Detection System, IDS），域名服务（Domain Name Service, DNS）等网络功能从专用的硬件设备中分离出来，以软件的形式在虚拟机中运行和实现。同时网络功能虚拟化建立在虚拟机技术之上（本文的模型与实现也与此相关），并扩展了其在网络领域的应用范围<sup>[24]</sup>。

网络功能虚拟化的应用价值主要但不仅限于如下三个方面<sup>[25]</sup>：

(1) 对于云计算而言，网络功能虚拟化允许服务供应商通过基于用户访问模式，用户移动性，基础架构负载特性，基础架构故障以及许多可能导致服务降级，中断或搅动的情况动态更改其部署拓扑或流量分配，从而为用户提供更好的服务。诸如 CloudNFV, CLOUDBAND 等重要先驱工作将网络功能虚拟化应用到云端。

(2) 对于移动网络而言，网络功能虚拟化使移动服务提供商拥有的数据中心受益<sup>[26]</sup>，其中包括移动核心网络，接入网络和移动云网络。它允许蜂窝提供商采用类似于数据中心的网络，该网络由结构简单的转发设备组成，大部分功能在靠近基站的商品服务器中执行。某些网络功能甚至可以通过直接安装在交换机<sup>[27-29]</sup>中的数据包处理规则来实现。在系统中，逻辑上集中的控制器能够通过所需的网络功能引导网络流量，以实现服务链。

(3) 对于企业级别的网络而言，网络管理员希望消耗所需的网络数量，但是企业客户想要的东西与服务提供商今天可以提供的东西之间存在差距，这可以通过该技术虚拟化解决。它可以在数分钟而不是数月内在商品服务器上动态配置虚拟网络服务。面向企业的网络功能虚拟化还支持 L4-7 软件以及其运营模式的变化。

#### 1.4 网络功能虚拟化的发展挑战与本文创新点

欧洲电信标准协会-网络功能虚拟化行业规范组涉及 200 多家公司，包括网络运营商，电信设备供应商，IT 供应商和技术提供商。该小组的创建是为了解决与虚拟化网络功能过程相关的各种挑战<sup>[30]</sup>。该小组定义的这些技术挑战包括：(1) 简化网络控制操作。(2) 通过虚拟化网络实现高性能。(3) 便携式工作器具。(4) 实现基于网络功能虚拟化的新型解决方案与基于硬件的传统网络平台的共存。(5) 虚拟网络设备的管理和编排，同时确保免受攻击和错误配置的安全。(6) 保持网络稳定性和服务水平，并且在设备加载和重新安置期间不会降低性能。(7) 确保对硬件和软件故障具有适当的恢复能力。(7) 减少能耗。

就本文边缘网络虚拟功能管理而言，有效部署虚拟网络功能实例需要解决几个关键挑战。在虚拟网络功能服务链部署异构网络功能时，供应商面临着不同目标之间的一些折衷，例如最小化网络的等待时间和最小化网络中活动节点的数量。这些目标相互矛盾，

因为在合并策略中最小化活动节点的数量会增加物理链路和结点上的聚合流量<sup>[31]</sup>，从而导致网络延迟目标<sup>[32]</sup>效率低下。同时，由于花费更多资源来部署虚拟网络功能实例，优化网络延迟会增加系统的成本。此外，当将多个虚拟网络功能服务链一起请求时，网络供应商不能忽略它们之间可能的依赖性。仅依靠资源使用的合并可能会导致物理网络拥塞，因为最小化活动结点点的数量会增加物理链路上所有嵌入式服务链的额外使用带宽<sup>[33]</sup>。而且在云上部署虚拟网络功能时，整合策略还面临着巨大的迁移成本，包括迁移过程成本（例如在 OpenStack 云中使用），重新配置系统或网络拥塞开销。有关目前解决上述网络功能虚拟化挑战的学术解决方案在第二章的文献综述中将被提及。

基于上述观察，在本文的研究中，我们考虑了服务链的虚拟网络功能调度问题来提高边缘网络虚拟功能的管理。我们假设虚拟网络功能（VNF）放置在网络中，并且可以由不同的服务共享。我们通过考虑 VNF 在虚拟机上的处理延迟和在虚拟链路上的服务链传输延迟来制定可感知延迟的 VNF 调度问题。同时，我们也更加符合实际，考虑在虚拟机容量和处理能力允许的范围内，允许在同一个时隙（time slot）内，一个虚拟机可以同时处理多个 VNF，从而最大程度地发挥虚拟机处理虚拟网络功能实例的实力，并通过扩展解决非合作博弈论中经典的稳定婚姻问题的 Gale-Shapley 算法<sup>[34]</sup>，提出分布式联盟算法（decentralized coalition algorithm），证实了在本文模型和算法中存在纳什均衡（Nash Equilibrium），实现满足所有网络约束的稳定调度，并且所有物理结点和 VNF 服务链都对该调度“满意”。

## 1.5 本文的组织结构

本文剩余章节安排如下。第 2 章我们阅读与调查了部分与 VNF 调度和资源分配问题有关的工作。第 3 章呈现了针对本文所研究问题而提出的模型与问题的定义。第 4 章将详细介绍本文主要创新性成果—分布式联盟算法并证实了纳什均衡的存在。第 5 章，陈述实验数值结果并分析模型特点。最后，总结全文，并提出未来的改进方案和工作目标。

## 2 网络功能虚拟化文献综述

随着 2012 年 10 月在德国达姆施塔特召开的软件定义网络（SDN）和 OpenFlow 会议，欧洲电信标准协会（ETSI）的规范小组出版了一份白皮书<sup>[35]</sup>，该白皮书介绍了在商用硬件上运行的虚拟化网络功能。国际互联网工程任务组（IETF）在 2018 年的草案提出了跨数据中心应用服务链的用例<sup>[36-37]</sup>，并强调了服务链与 NFV 之间的关系。NFV 的早期工作着眼于缩小专用硬件和网络功能之间的差距<sup>[38-40]</sup>，并为在虚拟机上实现网络功能提供了工业标准。工业界和学术界的研究活动也紧随其后。

经过阅读大量文献，本章将详细介绍虚拟网络功能的部署与动态网络功能链和虚拟网络功能服务链的调度两个类别网络功能虚拟化在学术方面的研究，着重介绍其结构特点和效果，以作为文献综述呈现。

### 2.1 虚拟网络功能的部署与动态网络功能服务链

目前大部分的关注点和学术研究主要集中在虚拟网络功能（VNF）的部署和动态网络功能链上，即 VNF 能需要在给定有限的网络资源下被高效地分配到物理结点并且在物理结点上映射到虚拟链路从而降低所需成本。通常需要降低成本大致为<sup>[41]</sup>

$$\min \sum_{t \in [T]} C_R^{(t)} + C_D^{(t)} + C_T^{(t)} + C_E^{(t)} \quad (2.1)$$

其中  $C_R^{(t)}$  为 VNF 的运行（Running）成本， $C_D^{(t)}$  为 VNF 的部署（Deployment）成本， $C_T^{(t)}$  为流量传输（Flow Transfer）成本， $C_E^{(t)}$  为端对端的流量时延（End-to-End Flow Delay）。

比如说，Wang, Wu, Le 等学者<sup>[42]</sup>采取了基于滑雪板租赁算法（Ski-rental Algorithm）的在线学习算法（online learning algorithm）和启发式算法（heuristic algorithm）来解决在一个云端数据中心中动态配置 VNF 的问题。具体表现为在给定构成企业网络服务的一个或多个服务链时，如何设计可动态扩展 VNF 实例并将其打包到服务器上的在线解决方案，以充分服务波动的输入流量，从而实现接近离线的最小流量在系统的长期运行中配置成本。（1）在单个服务链的情况下，作者提出了一种随机在线算法，该算法得出总成本的解决方案最多为离线最优解决方案的  $e/(e - 1)$  倍。（2）在有多个并发服务链的情况下，作者提出一种启发式算法，该算法依靠最小权重匹配算法来最小化部署成本并实现  $O(1)$  的竞争比率（competitive ratio）。

Jia, Wu, Li 等学者<sup>[41]</sup>研究了跨地理-分布式数据中心的 VNF 服务链的动态部署，以服务于分散的服务链的源（source）和目标（destination）组成的对之间的网络流量。（1）

作者制定了在线成本最小化问题，以实现在不同数据中心中动态部署和删除 VNF 实例，以及在各个服务链中的 VNF 实例之间进行动态业务流路由。作为网络服务供应的关键 QoS 性能指标，流量的平均端到端延迟也被制定并最小化。（2）作者使用在线学习文献中基于正则化的技术将整数离线优化问题（offline optimization problem）的松弛转化为一系列正则子问题（regularized sub-problems）。（3）最后作者设计了一个在线随机相关的舍入方案，用于将分数解（根据每个数据中心中 VNF 实例的数量）舍入为原始问题的可行整数解。其整体模型如图 2.1 所示。

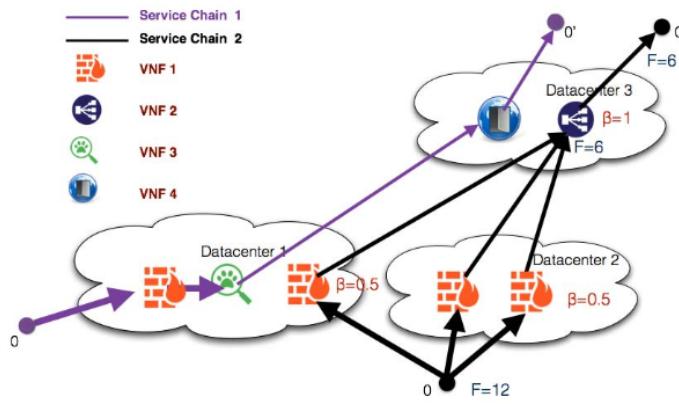


图 2.1 一个 NFV 服务链部署在跨地理分布的数据中心上

Cao, Fahmy, Sharma 等学者<sup>[43]</sup>提出了一种用于网络功能虚拟化的弹性资源伸缩系统（Elastic resource flexing system for NFV, ENVI），该系统利用 VNF 级别功能和基础架构级别功能的组合来构建基于神经网络的扩展决策引擎，以生成及时的扩展决策。同时作者开发了基于窗口的样本选择机制以及倒带（rewinding）机制，重新标记和选择两类样本的平衡数量，以便在观察到错误决策时训练当前的神经网络。图 2.2 和图 2.3 分别展现了 ENVI 的设计结构以及基于窗口倒带的例子。

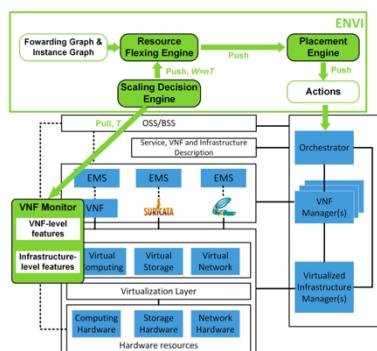


图 2.2 ENVI 的设计结构

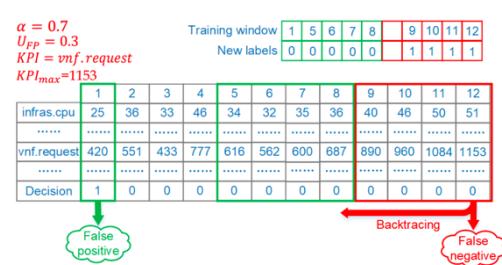


图 2.3 一个基于窗口倒带的例子

Xu, Liang 和 Galis 等学者<sup>[52]</sup>假设在事先已经在数据中心实例化有限数量的服务链实例的前提下进行支持 NFV 的网络中的吞吐量最大化问题，旨在接纳尽可能多的用户请求，同时将其实施成本降至最低并满足其端到端延迟要求。为了满足上述要求，作者提出了两种具有可证明的近似比率的近似算法（approximation algorithms，Appro-Split 与 Appro-Unsplit），并为该问题提供性能保证。图 2.4 阐明了作者为了实现整体最小成本和最大流量请求而构造的辅助图，基于该辅助图，作者提出了上述的近似算法。

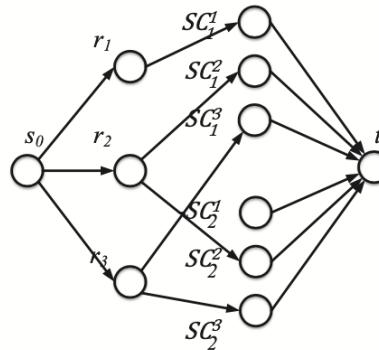


图 2.4 基于具有两个数据中心  $DC_1$  和  $DC_2$  以及三个要接受的请求（即  $r_1$ ,  $r_2$  和  $r_3$ ）的网络构造的辅助图  $G'$

Oechsner 和 Ripke<sup>[53]</sup>以自动化和透明的方式为 VNF 部署提供支持，该方式支持弹性模式和性能优化，旨在通过提供一种现实的解决方案来指定放置实例的基础架构的各个部分，并使用该名称在不同的优化步骤之间进行通信，从而使放置过程更加灵活和模块化。因此，关键基础架构组件应以确保高可用性或高性能的方式实例化，同时不要求服务运营商进行大量的手动规划和干预。为此，作者提出一种机制，根据适应的区域概念组织基础架构，从而为布局算法提供灵活的自由度。作者还提供了一个示例，说明如何在 OpenStack 中使用此方案，如图 2.5 所示。

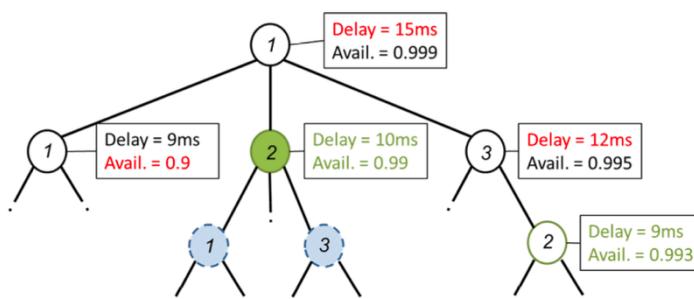


图 2.5 放置算法的说明，节点 1.2 和 1.3.2 都符合延迟和可用性要求，但是由于 1.2 提供了更高的自由度，因此它是首选方法，并且返回其两个子节点（带阴影的节点 1.2.1 和 1.2.3）

其他先前的工作（例如<sup>[44-46]</sup>）将 VNF 放置模型建模为整数线性规划（Integer Linear Programming, ILP），从而在网络流量和物理网络拓扑的约束下将部署和分配的总成本降至最低。

与此同时，也有一些研究使用博弈论来优化 VNF 的部署问题。

Leivadeas, Falkner 和 Lambadaris<sup>[47]</sup>在 2017 年 TNSM 上提出了一种图分割博弈 (graph partitioning game) 启发式算法以提高在满足服务器关联性，并置和时延等约束的云环境中部署服务链的整体分配性能。详细地说，(1) 引入多目标放置框架，其目标是最大程度地降低总体部署成本，该成本是 VNF 的计算和连接负载的总和。(2) 通过制定分割博弈，在给定可用公共云资源池的情况下证明作者设计的 VNF 服务链部署的最优性。(3) 提出一个受分割博弈的启发图形分区启发式算法，并适当调整其组成部分以优化最终解决方案。

Chen, Zhu, Guo 等学者<sup>[48]</sup>假设每个租户（tenant）的利润都与资源消耗成本和所实现的端到端服务等待时间有关，并提出了一种混合策略博弈（Mixed-Strategy Gaming）的方法，供租户在博弈中找到近似均衡解。为了激励租户更合理地使用网络资源，作者进一步设计了基于经纪人的弹性光数据中心网络（broker-based EO-IDCN）的动态资源定价策略，可以根据实际网络状态实时设置网络资源的价格，并验证了所提出的博弈论方法和动态定价策略的有效性。图 2.6 展现了 broker-based EO-IDCN 示例，图 2.7 说明了 EO-IDCN 中激励驱动的 VNF 服务链设置示例（a. 拓扑与供给策略，b. 对于 VNF 服务链的光谱部署）。

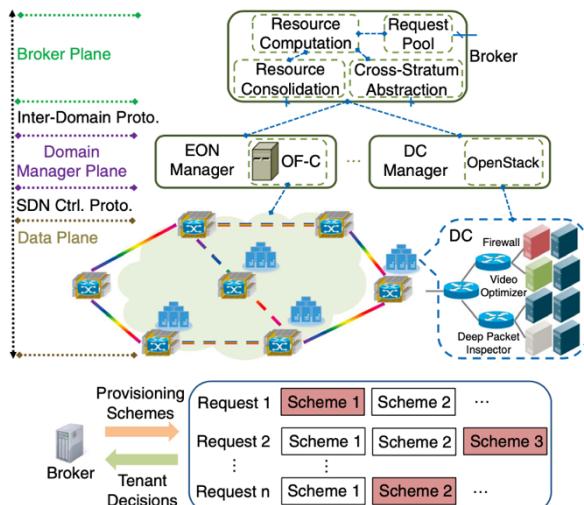


图 2.6 broker-based EO-IDCN 示例

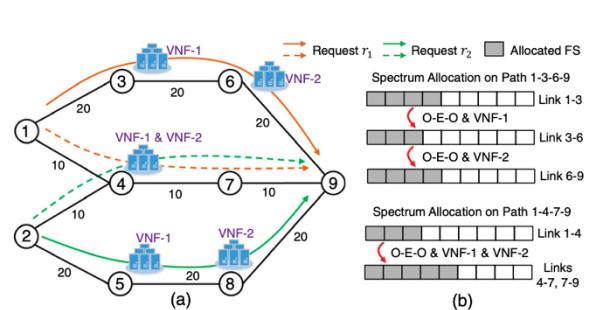


图 2.7 EO-IDCN 中激励驱动的 VNF 服务链设置示例

## 2.2 虚拟网络功能服务链的调度

相比于大量的虚拟网络功能（VNF）部署等相关方面的研究，在 VNF 服务链的调度问题上，研究数量就明显偏少。虚拟网络功能服务链的调度问题为在给定计算资源约束和特定服务的网络功能执行顺序的情况下，要求在相应的服务器/虚拟机上查找沿服务链的每个 VNF 的执行时隙。

Riera, Escalona, Bataille 等学者<sup>[30]</sup>考虑到 SDN 和 NFV 在光网络上的状态及其特定的限制，提供了对 SDN 和 NFV 方法的概述。据作者所知，他们是第一个形式化使用灵活的车间调度问题（Job-shop Scheduling Problem, JSP）来提供了 VNF 调度问题，为正式解决该问题提供了基础。最后，作者还提供了虚拟网络功能方法的概念验证原型。

Ma, Zhang 和 Huang 等学者<sup>[49]</sup>在 2017 年的 INFOCOM 上设计了一个集中式调度算法（centralized scheduling algorithm）来解决如下两个关键问题。（1）服务链调度。总体而言，没有具体确定服务链经过哪个结点。服务结点可以重用于多个 VNF。因此，可以在服务器上动态调度服务链的流程，以找到最佳的处理时隙，从而实现资源的高利用率。在此过程中，每个服务的逻辑服务链已固定，问题被简化为调度问题。（2）结点故障。每个服务节点都可能发生拥塞或故障，尤其是在资源利用率接近饱和时。一旦突发流量或许多请求到达，不稳定将引入到服务链中，包括很多数据包丢失，处理延迟，甚至服务中断以及许多其他问题。因此，在资源分配中应考虑可靠的机制，并尽可能避免由延迟或中断引起的不良用户体验，该文的 VNF 处理模型如图 2.8 所示。

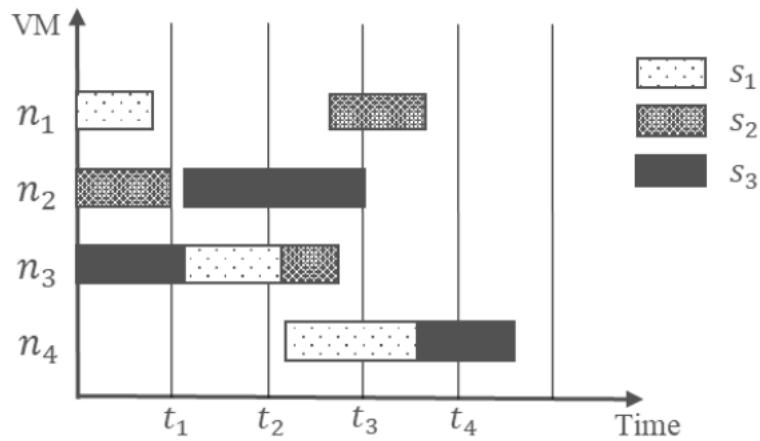


图 2.8 网络处理示意图

Qu, Assi 和 Shaban<sup>[50]</sup>研究了具有传输延迟优化的网络功能虚拟化调度的重要问题，并在 2016 年的 TOC 上提出了最佳的混合整数线性规划(Mixed Integer Linear Program,

MILP)框架和基于基因遗传 (genetic algorithm, GA) 的启发式算法以降低复杂度。此外，作者还探讨了动态虚拟链路带宽分配作为提高网络性能和比特率变化效果的有效方法的好处。结果表明，通过对连接虚拟机的虚拟链路动态分配带宽以及通过压缩操作分配计算资源，可以获得较短的计划。缩短时间表的重要性使服务提供商能够在其云数据中心接受并服务更多的流量，从而增加收入。VNF 调度示例如图 2.9 所示

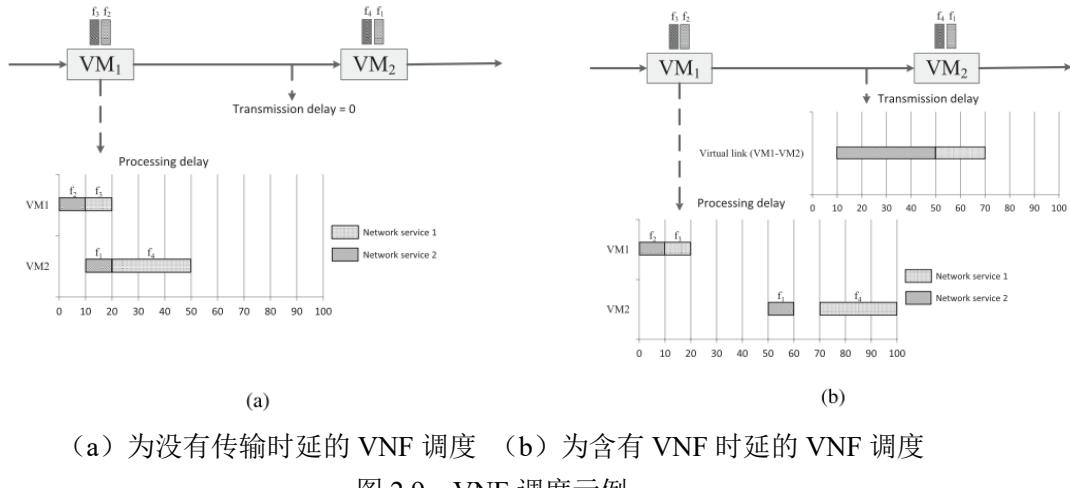


图 2.9 VNF 调度示例

Pham, Tran 和 Hong<sup>[51]</sup>提出将 VNF 调度问题理解为组合优化问题，该组合优化问题无法在非确定性多项式时间内找到最优解。为了解决这个问题，作者随后提出一种基于匹配的算法，以在每个时隙将 VNF 匹配到资源节点。此外，这种方法可以从离线 VNF 计划扩展到在线 VNF 计划，在这种情况下，控制器不需要知道未来的网络服务到达率。实验结果表明，所提出的基于匹配的方法可以达到最优结果的 87.2%。与循环法相比，该方法可以使完成的 VNF 数量增加 36.8%。VNF 调度和匹配方法分别如图 2.10 与图 2.11 所示。

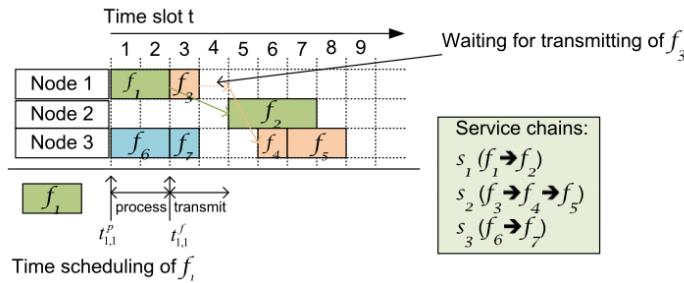


图 2.10 含有三条 VNF 服务链在三个物理结点上调度的示例

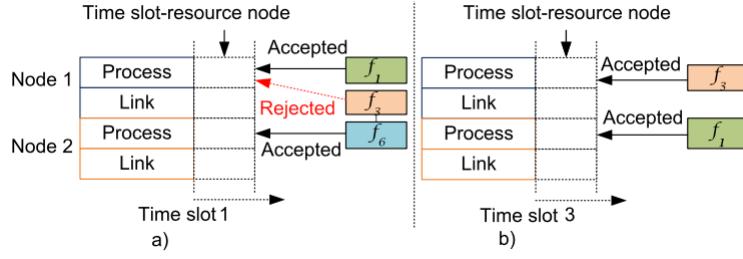


图 2.11 基于匹配方法的 VNF 调度在时隙 1 和时隙 3 时的状态

然而, [30], [49], [50]和[51]将 VNF 调度问题考虑每个虚拟机只能在同一个时隙处理一个 VNF 实例, 因此不适合进行多处理, 也与现实情况不符。

## 2.3 本章小结

本章主要阐述了虚拟网络功能的部署与动态网络功能链和虚拟网络功能服务链的调度, 这两方面的文献综述。图文并茂地介绍了在该任务上研究者们设计新算法或将已有算法应用到该领域中, 并揭示了每个算法所带来的的性能提升背后的机理与特点。这些内容有助于更好地将网络功能虚拟化, 这一新型研究领域落实到工业与实际生活应用中。本章内容作为后文研究的基础, 故单独将其置为一章。

### 3 基于博弈论的虚拟网络功能管理的模型建立

此部分，我们首先介绍系统模型和重要参数，接着详细地定义了问题模型，并说明该问题是 NP-hard。

#### 3.1 系统模型

我们考虑了一个由云服务器供应商运营的网络  $G = (DC, E)$ 。其中  $DC$  表示数据中心的集合， $E$  表示连接数据中心的链路的集合。每一个数据中心都有有限的计算和存储资源，从而在虚拟机中的软件上实施网络功能，即虚拟网络功能（VNF）。此外，我们假设在该基础设施的系统模型中有  $N$  个虚拟机和  $L$  条链路，这些虚拟机可以被分配到一个数据中心上或者可以在跨多个数据中心上来分配。任何两个虚拟机结点  $n$  ( $n \in [1, N]$ ) 都被虚拟链路  $l_n$  所连接。这里，我们引入符号  $P_n$  来表示每一个虚拟机  $n$  的处理容量，即每一个虚拟机  $n$  在同一时间最多可以同时处理 VNF 的个数。图一为我们系统模型的一个简单案例。

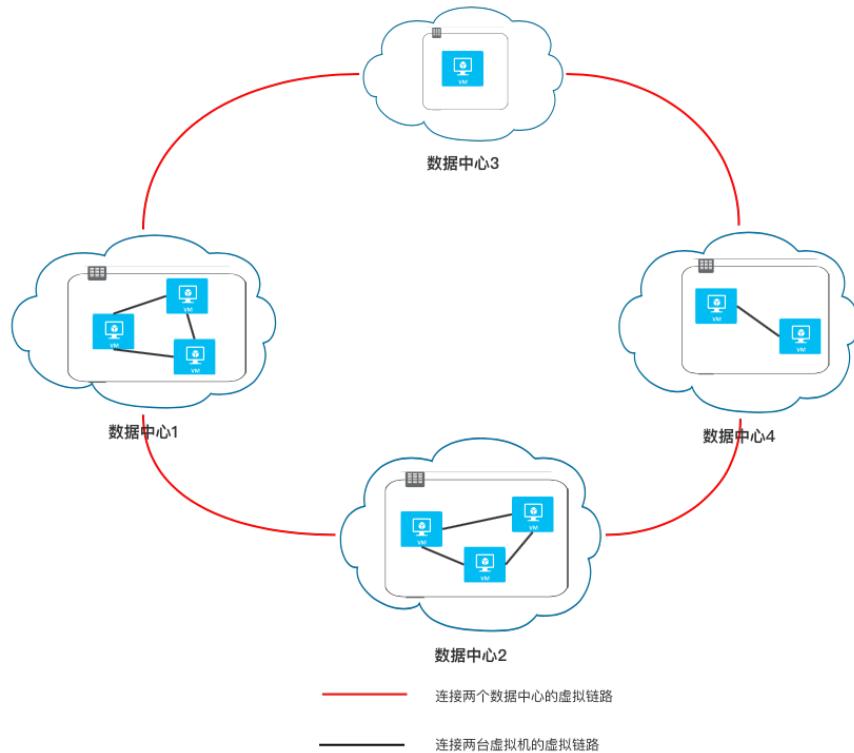


图 3.1 一个支持网络功能虚拟化的网络  $G$ ，其中有包含了多个虚拟机在内的数据中心，数据中心和虚拟机分别由虚拟链路  $E$  和  $l_n$  来连接

### 3.2 虚拟网络功能服务链

一个虚拟网络功能（VNF）服务链包含数个 VNF 实例。通常情况下，一条服务链当中每个 VNF 实例之间都存在相互依赖的关系。对于特定应用程序中的某条特定数据流，服务供应商通过多个 VNF 管控网络流量，从而完成特定的网络功能<sup>[24]</sup>。用  $VNF_s$  表示一条服务链的源（source），表示通过服务链的流量总来源。类似地， $VNF_d$  也被创建来表示一条服务链的目的（destination）。所以  $VNF_1$  才是该服务链中的第一个 VNF 实例，并且  $VNF_2, VNF_3, \dots, VNF_{j-1}, VNF_j$  被有序地部署在数据中心中的虚拟机上（比如说，防火墙通常被部署在整个服务链的始端来确保整个服务链的安全）。

在本模型中，有一系列的网络服务  $S$  被提供。对于每一个网络服务链  $i \in S$ ，都有一系列 ( $F$ ) 的 VNF 实例需要根据顺序在虚拟机上被处理或者在虚拟机之间相互传输。因此，我们定义了  $f_{ij}$  来表示在第  $i$  条服务链上面的第  $j$  个 VNF 实例 ( $i \in S, j \in F$ )。我们引进变量  $R_i$  ( $i \in S$ ) 来假设每一个网络服务是带有特定需求的，需要从源节点传输到目的结点的网络流量。

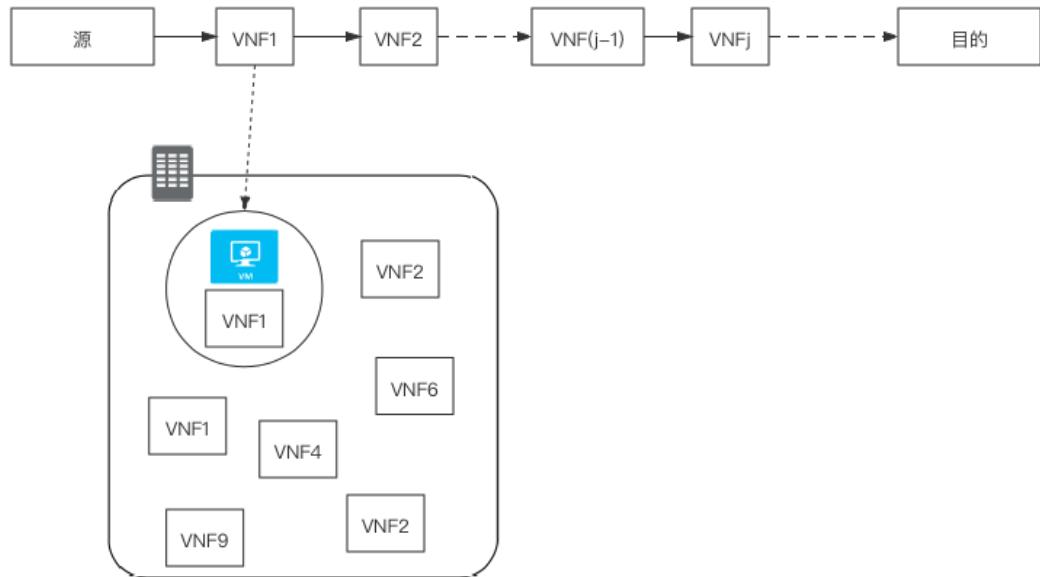


图 3.2 一条嵌入在支持 NFV 网络中的 VNF 服务链

### 3.3 问题定义

本模型的目标是找到虚拟网络功能实例和虚拟机结点之间存在的均衡策略，即找到在对于每一个即将到来的服务结点上的服务链流量中适当的 VNF 处理和传输时延。因

此，资源使用和网络服务的完成时间被当做是优化的目标。用 $x_{ij}$ 来表示处理服务链*i*上第*j*个VNF实例的起始时间。同时，在虚拟机上处理一个VNF实例需要一定的处理时延，所以我们引进变量 $p_{nf_{ij}}$ 来表示运行在虚拟机*n*上的第*i*条服务链中的第*j*个VNF实例。对于VNF调度而言，让二进制变量 $\theta_{ijn}^p$ 来描述在虚拟机结点上流量的分配，其中如果第*i*条服务链中的第*j*个VNF实例被分配在了虚拟机*n*上的话， $\theta_{ijn}^p = 1$ ，反之亦然。此外，整个系统模型按照时隙（time slot）来运行。因此，服务完成时间以最后一条服务链的处理结束而结束，可以表示为

$$\lambda = \max_{i \in S, j \in F} (x_{ij} + \sum_{n \in N} \theta_{ijn}^p p_{nf_{ij}}) \quad (3.1)$$

我们的模型同时也考虑了传输时延。使变量 $y_{ij}$ 表示在服务链*i*上开始从第*j*个VNF实例到第*j+1*个VNF实例流量数据包的起始时间。下一个VNF实例的网络服务的流量只能在通过连接两个虚拟机的虚拟链路完成传输后，才可以继续被执行。相似地，我们引进二进制变量 $\theta_{ijl_n}^t$ 来描述来自承载服务链*i*的第*j*个VNF流量的虚拟机的传出虚拟链路*l\_n*的分配，其中当且仅当服务链*i*上的第*j*条虚拟链路被映射在虚拟机上的时候， $\theta_{ijl_n}^t = 1$ 。此外，参数 $C_l$ 是连接两个虚拟机*n*与*n'*的虚拟链路*l\_n*的带宽。所以，在虚拟链路*l\_n*上的 $f_{ij}$ 的传输时延可以表示为 $\frac{R_i}{C_l}$ 。图3.3阐明了一个关于VNF实例如何被处理和传输的案例。

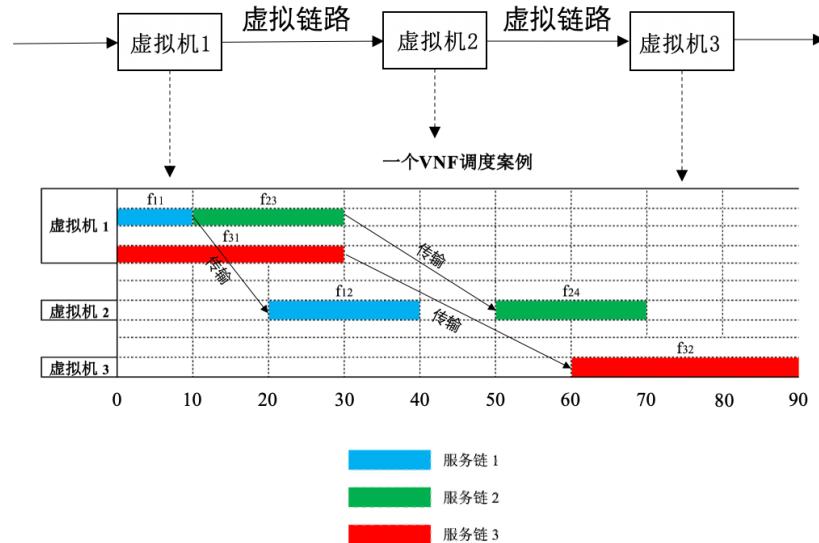


图3.3 一个含有三个虚拟机（虚拟机1，虚拟机2，虚拟机3）和三条VNF服务链（服务链1，服务链2，服务链3）的VNF调度示例

考虑图 3.3 中的三条服务链（服务链 1 上有  $f_{11}$  和  $f_{12}$  两个 VNF 实例，服务链 2 上有  $f_{23}$  和  $f_{24}$  两个 VNF 实例，服务链 3 上有  $f_{31}$  和  $f_{32}$  两个实例）。我们假设连接虚拟机 1，虚拟机 2 和虚拟机 3 的带宽是 1Mbps，同时假设服务链 1，服务链 2 和服务链 3 的流量请求  $(R_1, R_2, R_3)$  分别为 10kb，20kb 和 30kb。因此，回顾传输时延的公式  $\frac{R_i}{C_l}$ ，服务链 1，服务链 2，服务链 3 所对应的传输时延分别为 10ms，20ms，30ms。此外， $f_{11}$  和  $f_{31}$  在满足虚拟机 1 处理能力的前提下可以同时在虚拟机 1 上被处理，同理  $f_{23}$  和  $f_{31}$  也是一样。

在描述模型的条件现值之前，我们先列出了一些合理的假设：

(1) 每一个虚拟机在时隙  $t \in T$  期间只能处理一定数量的 VNF 实例，这受限于虚拟机的处理能力  $P_n$  并且其他的 VNF 实例需要等待直到前指定的虚拟机完成了相应的处理工作。定义一个二进制变量  $\delta_{ijn}^p$  来表示服务链  $i$  上第  $j$  个 VNF 实例是否正在被虚拟机结点  $n$  处理。如果  $\delta_{ijn}^p = 1$ ，则表示正在被处理，反之亦然。相似地，只有当前序 VNF 实例被传输完成之后，下一个 VNF 结点才可以被处理（可以参考图 3.3）。再定义一个二进制变量  $\delta_{ijl_n}^t$  表示服务链  $i$  上第  $j$  个 VNF 实例是否正在虚拟链路  $l_n$  上上传输。

(2) 每一条服务链上的 VNF 实例在同一个时隙  $t$  上只能被部署在一个虚拟机上并且一些 VNF 实例不能在一个虚拟机上面被并列放置，因为他们有可能会负面影响彼此的性能，即在相同虚拟机上面的服务链不能够相互冲突。

(3) 为了避免一种情况，即一个 VNF 实例长时间在被处理，从而其他的 VNF 实例不能使用该虚拟机或者影响该 VNF 的传输导致同一条服务链中的下一个 VNF 实例不能被有效处理。我们引入强制性截止时间  $dl_{ij}$  来表示  $f_{ij}$  的最大处理时延。然而，当一个 VNF 实例在虚拟机上正常被处理时（小于规定的最大处理时延），在规定的时隙内不允许被中断。

(4) 并不是所有的虚拟机都可以提供必要的硬件需求来承载特定的 VNF 实例。

下列的条件约束确保了一条 VNF 服务链的处理和传输顺序：

$$\sum_{i \in S, j \in F} \theta_{ijn}^p \delta_{ijn}^p \leq P_n \quad (3.2)$$

$$x_{ij} + \theta_{ijn}^p \delta_{ijn}^p \leq dl_{ij} \quad (3.3)$$

$$x_{ij} + \sum_{n \in N} \theta_{ijn}^p \delta_{ijn}^p p_{nf_{ij}} \leq y_{ij} \quad (3.4)$$

$$y_{ij} + \sum_{l_n \in L} \theta_{ijl_n}^t \delta_{ijl_n}^t \frac{R_i}{C_l} \leq x_{ij+1} \quad (3.5)$$

$$\sum_{i \in S, j \in F, l_n \in L} \theta_{ijl_n}^t \delta_{ijl_n}^t \leq 1 \quad (3.6)$$

$$\sum_{n \in N} \delta_{ijn}^p = 1 \ (\forall i \in S, j \in F) \quad (3.7)$$

$$\sum_{n \in N} \delta_{ijl_n}^t = 1 \ (\forall i \in S, j \in F) \quad (3.8)$$

约束 (3.2) 确保了在相同时隙  $t$  时, 一个虚拟机有足够的能力来处理多 VNF 实例但是在虚拟机  $n$  上处理的 VNF 实例的数量需要满足虚拟机处理能力  $P_n$  的要求。约束 (3.3) 确保了处理一个 VNF 实例的硬性结束时间。 (3.4) 保证了第  $j$  个 VNF 实例的传输转发时必须得等到虚拟机  $n$  完成该 VNF 实例的流量处理工作, 同时 (3.5) 保证了除非下游流量通过连接两个虚拟机的虚拟链路  $l_n$  从上游虚拟机转发, 否则下游虚拟机上面的第  $j + 1$  个 VNF 实例的处理不应该开始。总体来说, (3.4) 和 (3.5) 保证了 VNF 处理和传输的顺序要求。约束 (3.6) 表明了在每一个时隙  $t$ , 虚拟链路  $l_n$  只能传输一个 VNF 实例。

(3.7) 确保了一个 VNF 实例在时隙  $t$  时只能在一个虚拟机而不是多个虚拟机上面被处理。 (3.8) 保证了至多一条虚拟链路可以被选择来传输一个 VNF 实例, 这表明了一个 VNF 实例不可以被拆分同时在多条虚拟链路上被传输。

为了方便参考, 我们在表 3.1 中列出了重要参数。

表 3.1 重要参数

$DC$	# 数据中心的集合
$E$	# 连接数据中心的链路的集合
$N$	# 在该基础架构中虚拟机的集合
$L$	# 在该基础架构中连接两台虚拟机的虚拟链路的集合
$S$	# 网络服务链的集合
$F$	# VNF 实例的集合
$T$	# 时隙的集合
$f_{ij}$	在服务链 $i$ 上的第 $j$ 个 VNF 实例
$R_i$	服务链 $i$ 的流量请求
$C_l$	虚拟链路 $l_n$ 的带宽
$P_n$	每一个虚拟机 $n$ 的处理能力/容量
$x_{ij}$	服务链 $i$ 上的第 $j$ 个 VNF 实例开始处理的起始时间
$y_{ij}$	服务链 $i$ 上第 $j$ 个 VNF 实例和第 $j + 1$ 个 VNF 实例之间开始转发传输的起始时间
$p_{nf_{ij}}$	服务链 $i$ 上第 $j$ 个 VNF 实例的处理时延
$\theta_{ijn}^p$	一个二进制变量来描述网络功能是否被布置到了虚拟机上

续表 3.1 重要参数

$\theta_{ijl_n}^t$	一个二进制变量用于描述网络功能是否在虚拟机的传出虚拟链接 $l_n$ 上
$\delta_{ijn}^p$	一个二进制变量来描述服务链 $i$ 上第 $j$ 个 VNF 实例是否正在被虚拟机 $n$ 处理
$\delta_{ijl_n}^t$	一个二进制变量来描述服务链 $i$ 上第 $j$ 个 VNF 实例是否正在虚拟链路 $l_n$ 上被传输
$dl_{ij}$	服务链 $i$ 上第 $j$ 个 VNF 实例处理的强制截止时间

作为结论，VNF 的调度问题可以被定义如下：

$$\min \lambda$$

满足约束 (3.2) - (3.8)

定理 1：该 VNF 调度问题 (3.1) 是非确定性多项式困难问题 (NP-hard)。

证明 1：与 VNF 调度问题是“车间调度问题” (Job-shop Scheduling Problem, JSP) 的相关问题<sup>[30]</sup>，而车间调度问题被认为是 NP-hard。在车间调度问题中，需要  $m$  ( $m \in N^*$ ) 个机器需要处理  $n$  ( $n \in N^*$ ) 件半制品。每一件半制品都需要特定的处理技术并且半制品的处理顺序和所需要的处理时间被提供。车间调度问题的目标是决定每一个机器上处理顺序和每一次处理的起始时间，使得总处理时间最小或者优化其他的指标。因此，我们定义的 VNF 调度问题的难度至少和车间调度问题一样，且车间调度问题忽略了半制品的传输时间，而在 VNF 调度问题需要考虑 VNF 实例在虚拟机之间的传输时延。

### 3.4 本章小结

本章主要阐明了本文中模型的建立。首先阐明了网络拓扑的构建，即一个支持网络功能虚拟化的网络  $G$ ，含有跨地理位置的数据中心，在数据中心上部署着一定数量的虚拟机，虚拟机内运行着虚拟网络功能实例。其次，定义并说明 VNF 服务链的作用，并用图片的方式可视化服务链。最后定义问题，该问题是 NP-hard，在满足相关约束的前提下，要使整个 VNF 的完成时间最小，是该 VNF 调度的目标。同时，用图片的方式可视化 VNF 处理和传输的过程。为了解决该问题，我们在下一章将给出落实的算法并证明算法的可靠性。

## 4 分布式联盟构建机制

在本章，我们展现了分布式联盟构建（decentralized coalition formation）的过程。该过程基于对于“稳定婚姻问题”(Stable Marriage Problem)传统的延迟接受程序(deferred-acceptance procedure)，将 VNF 实例与虚拟机结点相匹配。在该过程中，VNF 实例扮演者请求“女生”的“男生”角色，虚拟机结点在该过程中是女生。因此，可以将我们模型中的“稳定婚姻问题”指定为在特定虚拟机容量的约束下，如何处理和传输理想数量的满足条件的 VNF 实例，而又不确切知道将接受多少个 VNF 实例。

### 4.1 虚拟网络功能实例与虚拟机的偏好列表

#### 4.1.1 虚拟网络功能实例的偏好列表

在服务链上每一个 VNF 实例  $f_{ij}$  在包含其自身的可行性联盟 (feasible coalition) 中有一个由  $\mu_{ij}(\cdot)$  定义的优先级列表 (preference list)。定义一个有序  $f_{ij}$  地偏好列表  $Pref_{f_{ij}}$  (例如,  $Pref_{f_{11}} = (VM_1, VM_2, \dots, VM_n, \dots, Null)$ )，从而使得  $\mu_{ij}(VM_{n-1}) \geq \mu_{ij}(VM_n)$ 。因为任何 VNF 实例都希望被尽快地处理或者传输，所以该偏好列表是基于时延而定的。为了列表的完整性，我们也加入了最不被偏好的选项  $Null$  在列表的末端，因此  $\mu_{ij}(Null) = -\infty$ 。虚拟网络功能实例的偏好列表伪代码如表 4.1 所示。

#### 4.1.2 虚拟机的偏好列表

相似地，在每一个时隙  $t$  时，在虚拟机上处理以及在虚拟机之间传输可以同时进行。我们引入虚拟机的偏好列表  $Pref_n$ ，该列表基于 VNF 实例的流量计算请求。即一个 VNF 处理/传输请求的计算资源越少时，该 VNF 实例在虚拟机的偏好列表中的位置越靠前。同理，为了保证列表的完整性，我们也加入了最不被偏好的选项  $Null$  在列表的末端，使得  $\mu_n(Null) = -\infty$ 。虚拟机的偏好列表伪代码如表 4.2 所示。

#### 4.1.3 阻塞组合与稳定联盟

**定义 1：**一组关于联盟 (coalition) 结构的 VNF 实例的集合  $M \subset F$  被成为阻塞组 (blocking group) 如果所有在  $M$  中的 VNF 实例在其他组合中可以改善他们的处理/传输时延的话。一个组合可以被成为稳定联盟结构 (stable coalition structure) 如果该组合中不存在阻塞组，即如果将该组合中的每个 VNF 实例转移至其他的联盟，VNF 实例的表现都会变差。同时一个联盟是不稳定的 (unstable coalition structure)，如果该联盟中的 VNF 实例在其他组合中能更好地完成处理/传输任务。

表 4.1 创建虚拟网络功能实例偏好列表的伪代码

---

**算法 4.1** 计算 VNF 实例的偏好列表

---

**输入:** 虚拟机  $n$ , VNF 实例  $f_{ij}$

**输出:**  $Pref_{f_{ij}}$

▷ 从当前 VNF 实例  $f_{ij}$  开始

1: **重复**

2:   **如果**  $f_{ij}$  在当前虚拟机  $n$  上有着更少的处理/传输时延 那么

3:     将  $n$  加入有序地加入到  $Pref_{f_{ij}}$

4:     将该虚拟机  $n$  标记为“已用”： $Pref_{f_{ij}} \leftarrow n$

5:      $n \leftarrow next(n)$

6: **结束如果条件**

7: **直到** 所有虚拟机  $n$  都在  $Pref_{f_{ij}}$  中

8: 清除虚拟机  $n$  的“已用”标记

9: 将  $Null$  加入到  $Pref_{f_{ij}}$  中

10:  $Pref_{f_{ij}} \leftarrow next(Pref_{f_{ij}})$  **直到** 所有  $Pref_{f_{ij}}$  都被遍历

11: ▷ 结束

12: 返回 VNF 偏好列表  $Pref_{f_{ij}}$

---

表 4.2 创建虚拟机偏好列表的伪代码

---

**算法 4.2** 计算虚拟机结点的偏好列表

---

**输入:** 虚拟机  $n$ , VNF 实例  $f_{ij}$

**输出:**  $Pref_n$

▷ 从当前虚拟机  $n$  开始

1: **重复**

2:   **如果**  $f_{ij}$  在当前虚拟机  $n$  上占用更少的计算资源 那么

3:     将  $f_{ij}$  加入有序地加入到  $Pref_n$

4:     将该 VNF 实例  $f_{ij}$  标记为“已用”： $Pref_n \leftarrow f_{ij}$

5:      $f_{ij} \leftarrow next(f_{ij})$

6: **结束如果条件**

7: **直到** 所有 VNF 实例  $f_{ij}$  都在  $Pref_n$  中

8: 清除虚拟机  $n$  的“已用”标记

9: 将  $Null$  加入到  $Pref_n$  中

10:  $Pref_n \leftarrow next(Pref_n)$  **直到** 所有  $Pref_n$  都被遍历

11: ▷ 结束

12: 返回虚拟机偏好列表  $Pref_n$

---

## 4.2 分布式联盟算法

此算法基于经典的 Gale-Shapley 算法进行拓展改编，考虑到分散的联合过程可能会导致不止一个稳定结果，具体取决于 VNF 实例是先请求虚拟机结点还是虚拟机结点先请求 VNF 实例，我们算法选择前项，即 VNF 实例请求虚拟机。并且我们初始化时隙  $t$  为 1。

在联盟构建的过程中，每个 VNF 实例和虚拟机结点含有一组变量：

$$(Props_n, Held_n, Suspended_{f_{ij}})$$

- (1)  $Props_n$  是虚拟机  $n$  收到的请求联盟组合。
- (2)  $Held_n$  是虚拟机  $n$  目前正在考虑的联盟。
- (3)  $Suspended_{f_{ij}}$  是一个布尔变量，表示 VNF 实例是否可以最求其自身偏好列表中下一个虚拟机结点。

表 4.3 为描述该分布式联盟算法的伪代码。

表 4.3 分布式联盟算法

---

### 算法 4.3 分布式联盟算法

---

**输入:**  $Pref_{f_{ij}}, Pref_n, Props_n, Held_n, Suspended_{f_{ij}}$

**输出:**  $Held_n$

▷ 初始化

1: 对于  $i \in S, j \in F, n \in N$  循环

2:    $Held_n \leftarrow Null, Suspended_{f_{ij}} \leftarrow False, Props_n \leftarrow \emptyset$

3: 结束循环

4: 重复

5:   对于  $Suspended_{f_{ij}} = False$  循环

6:      $Props_n \leftarrow Held_n \cup f_{ij}$

7:     ▷  $f_{ij}$  请求在  $Pref_{f_{ij}}$  中排名最高的虚拟机  $n$

8:   结束循环

9:   对于  $Props_n \neq \emptyset$  并且  $\sum_{i \in S, j \in F} Props_n > P_n$  循环

10:     ▷ 在  $Pref_n$  中寻找上述  $Props_n$  中最不被偏爱的 VNF 实例  $f_{ij}$

11:     如果  $\mu_n(f_{ij}) > \mu_n(f_{ij'}) \in Held_n$  那么

12:        $Held_n \leftarrow Props_n \setminus (f_{ij'})$

13:       ▷ 从  $Held_n$  中移除那些最不被偏爱的 VNF 实例

14:   结束如果条件

15:     $Pref_{f_{ij'}} \leftarrow Pref_{f_{ij'}} \setminus (n), Pref_n \leftarrow Pref_n \setminus (f_{ij'})$

---

续表 4.3 分布式联盟算法

---

```

16:     $\triangleright$  将  $f_{ij}$  和  $n$  分别从  $Pref_n$  和  $Pref_{f_{ij}}$  中移除
17:    结束循环
18:    对于  $f_{ij} \in Held_n$  循环
19:         $Suspended_{f_{ij}} \leftarrow True$ 
20:    结束循环
21: 直到  $f_{ij} \in Held_n$  对于所有的  $i \in S, j \in F, n \in N$ 
22:  $\triangleright$  处理/传输可以进行

```

---

$Top(Props_n)$  表示在  $Props_n$  中优先级最高的选项。最开始，我们初始化  $Held_n \leftarrow Null$ ,  $Suspended_{f_{ij}} \leftarrow False$ ,  $Props_n \leftarrow \emptyset$  (第 1-3 行)。VNF 实例请求可用的虚拟机结点 (第 5-8 行)，同时需要满足虚拟机结点的处理容量。每一个虚拟机扮演着“女生”的角色来接受不属于任何联盟的 VNF 实例发来的请求。接着虚拟机按照自身的偏好列表  $Pref_n$  开始比较收到请求的 VNF 实例的顺序，并在自身容量允许的范围内接受排名最高的 VNF 实例，同时放弃剩余/排名较低的 VNF 实例。被放弃的 VNF 实例也会将相应的虚拟机结点从其偏好列表  $Pref_{f_{ij}}$  中移除。与此同时，虚拟机也将这些 VNF 实例从其偏好列表中移除  $Pref_n$  (第 9-19 行)。当每一个 VNF 实例不再允许请求下一虚拟机时或者当所有虚拟机容量都达到最大值时，稳定联盟构建，此时即可开始 VNF 实例的处理。

为了方便理解，我们给出了一个简单案例。考虑三个虚拟机 (A, B, C) 和一条服务链上的五个 VNF 实例 (a, b, c, d, e) 且 A 与 B 虚拟机在同一时隙内最多可以同时处理两个 VNF 实例，C 在同一时隙内只能处理一个 VNF 实例，即  $P_A = P_B = 2, P_C = 1$ 。对于虚拟机和 VNF 实例的偏好列表如下表所示。

表 4.4 虚拟机与 VNF 实例的偏好列表

$Pref_n$			$Pref_{f_{ij}}$				
A	B	C	a	b	c	d	e
a	a	c	<b>A</b>	<b>C</b>	<b>B</b>	<b>B</b>	C
b	<b>c</b>	<b>b</b>	B	B	A	C	<b>A</b>
c	<b>d</b>	e	C	A	C	A	B
d	e	d					
<b>e</b>	b	a					

$Pref_1 = (a, b, d, c, e), Pref_2 = (a, c, d, e, b), Pref_3 = (c, b, e, d, a) \quad ; \quad Pref_{f_{11}} = (A, B, C), Pref_{f_{12}} = (B, A, C), Pref_{f_{13}} = (B, A, C), Pref_{f_{14}} = (B, C, A), Pref_{f_{15}} = (C, A, B)$ 。分布式联盟构建的过程如表 4.5 所示：

表 4.5 该示例中分布式联盟构建的过程

第一轮	
a → A	$Props_A = (a), Props_B = (c, d), Props_C = (b, e)$
b → C	$\because P_c = 1 \text{ and } \mu_c(b) > \mu_c(e)$
c → B	$\therefore Held_A = (a), Held_B = (c, d), Held_C = (b)$
d → B	$\therefore Suspended_a = Suspended_b = Suspended_c = Suspended_d = True$
e → C	$\therefore Suspended_e = False$

第二轮	
	$Props_A = (e)$
	$\because P_A = 2$
e → A	$\therefore Held_A = (a, e)$
	$\therefore Suspended_a = Suspended_b = Suspended_c = Suspended_d = Suspended_e = True$

经过分布式联盟算法之后得出的稳定联盟为  $(A \leftrightarrow a, e), (B \leftrightarrow c, d), (C \leftrightarrow b)$ 。值得一提的是，为了说明算法和分布式联盟构建的过程，我们给出了一个简单的案例，但在真实网络情况下，虚拟机和 VNF 实例较多，分布式联盟的构建远比该例子要复杂的多，具体的仿真实现细节将在第 5 章中呈现。

### 4.3 分布式联盟算法中的纳什均衡

定理 2：在分布式联盟算法中存在纳什均衡（Nash Equilibrium）。

证明 2：根据定义 1，我们假设  $(n, f_{ij}, f_{ij'}, \dots)$  是一个阻塞组（定义 1）。导致出现阻塞组有两种可能的情况。一种是  $f_{ij}$  这个 VNF 实例从来没有请求过虚拟机结点  $n$ 。在这请可能性下， $n$  在  $f_{ij}$  的偏好列表  $Pref_{f_{ij}}$  中排名较低，所以  $f_{ij}$  先请求了其他的虚拟机，但是被其他的虚拟机拒绝后  $f_{ij}$  在请求该虚拟机结点  $n$ 。另一种可能是  $f_{ij}$  已经请求虚拟机结点  $n$  但是被此虚拟机拒绝，在这种情况下，不论是 VNF 实例  $f_{ij}$  还是虚拟机  $n$  都将彼此从各自的偏好列表中移除。上述的两种情况都反证出联盟  $(n, f_{ij}, f_{ij'}, \dots)$  是一个稳定的联盟，这与假设的阻塞组相冲突。因此调度联盟  $(n, f_{ij}, f_{ij'}, \dots)$  得比调度其他联盟  $(n^*, f_{ij}^*, f_{ij'}, \dots)$  具有更加的表现和稳定性，这也证明了在该“分布式联盟算法”中存在着纳什均衡。

#### 4.4 本章小结

本章详细介绍了本文的创新性结果，“分布式联盟算法”用于解决 VNF 实例调度的问题，我们将该 NP-hard 问题抽象为“稳定婚姻问题”，且“分布式联盟算法”基于经典的处理稳定婚姻问题 Gale-Shapley 算法进行拓展改编，具有可靠的理论依据，将虚拟机与 VNF 实例进行联盟组合，从而得到稳定联盟结构，证实了纳什均衡的存在。与 Gale-Shapley 算法不同的是，Gale-Shapley 用于解决的“稳定婚姻问题”是一个一对一的匹配问题，而本文提出的“分布式联盟算法”适用于解决一对一或者一对多的匹配问题。为了说明分布式联盟构建的具体过程，我们给出了一个简单的案例，但实际的网络拓扑远比该案例复杂，虚拟机与 VNF 实例的数量更为庞大，分布式联盟的构建步骤更多，具体的实验结果与数值分析将在下章呈现。

## 5 实验结果及其数值分析

在本章，我们评估了在第 4 章中所设计的分布式联盟算法的虚拟网络功能调度问题的实验设定。接着详细地阐述了本文的测试结果，并通过分析讨论揭示该算法的特点。

### 5.1 实验环境

根据第 3 章定义的参数，变量和模型进行网络拓扑的构造，将网络功能和虚拟机的相关信息保存在 Excel 表格中，最后在 PyCharm IDE 上使用 Python 进行仿真，代码编写与图像绘制。

#### 5.1.1 NetworkX

本文使用 NetworkX 进行网络拓扑图的构建。NetworkX 在 2002 年五月由 Aric Hagberg, Dan Scult 和 Pieter Swart 设计与编写。它是一个 Python 软件包，用于创建，操作和研究复杂网络的结构，动力学和功能。在研究社会，生物和基础设施网络的结构和动态的工具。在网络拓扑应用中，NetworkX 可以以标准和非标准数据格式加载和存储网络，生成多种类型的随机和经典网络，分析网络结构，构建网络模型，设计新的网络算法，绘制网络等。我们使用该软件包生成第 3 章所构建的虚拟网络功能调度问题的模型与网络拓扑。

#### 5.1.2 Matplotlib

Matplotlib 是一个综合库，用于在 Python 中创建静态，动画和交互式可视化。它是一个用于在 Python 中制作数组二维图的库。尽管它起源于仿真 MATLAB 图形命令，但它独立于 MATLAB，可以以 Pythonic 的，面向对象的方式使用。尽管 Matplotlib 主要是用纯 Python 编写的，但它大量使用了 NumPy 和其他扩展代码，即使对于大型数组也可以提供良好的性能。为了形象地向读者描述第 3 章构建的网络拓扑图与系统模型，我们使用 Matplotlib 进行上述图像的可视化处理。

#### 5.1.3 xlrd

本实验的网络拓扑图相关数据全存放在 Excel 表格当中，在 Python 当中使用 xlrd 数据包进行对 Excel 表格数据的读取。

### 5.2 模型实现细节

我们将仿真了两种不同情况的实验。（1）我们首先构造了一个包含 30, 50 和 100 个虚拟机结点全联通网络拓扑图以处理 100 条 VNF 服务链。（2）我们又构造了一个含

有 50 个虚拟机结点全联通网络拓扑图以处理 10 条，20 条，30 条，40 条和 50 条 VNF 服务链。两种情况的仿真均满足以下条件：（1）虚拟链路的带宽位于 2-10 Mbps 之间，（2）VNF 实例的流量请求在 60-100 kbps 之间，（3）虚拟机处理单位 VNF 实例的请求在 3-6 ms 之间，所以每个 VNF 实例的处理时长大致在 10-30 ms 之间（4）每一条服务链上面都有 3-5 个 VNF 实例，（5）每个虚拟机在一个时隙上最多可以同时处理 3 个 VNF 实例。（6）VNF 为防火墙（Firewall），代理（Proxy），网络地址转换（NAT），入侵检测系统（IDS）和负载均衡（Load Balance）五个中的随机生成的一个或多个网络功能。同时，为了实验的严谨性，避免某单个实验出现特殊情况，对于其中任何一种情形的仿真，我们进行了 10 次随机实验，并记录实验结果，得出了一般普遍规律，实验结果将在本章后半段呈现。图 5.1 分别呈现了使用 NetworkX 生成 30 个虚拟机结点，50 个虚拟机结点和 100 个虚拟机结点的网络拓扑图（网络拓扑里面的虚拟机与虚拟链路的数量固定，但是拓扑图形状不固定）。

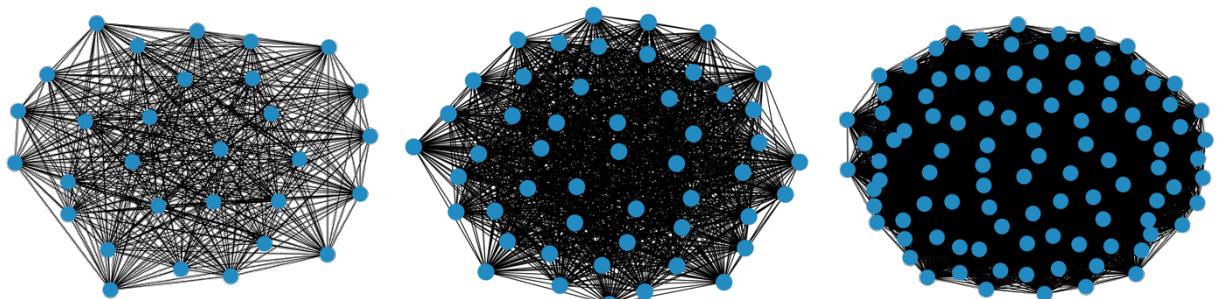


图 5.1 所用的网络拓扑图  
(分别为 30 个虚拟机，50 个虚拟机和 100 个虚拟机的全联通图)  
其中蓝色结点为虚拟机，黑色线为连接虚拟机的虚拟链路

表 5.1 展现了同一种情况下，仿真所用的上述 5 种 VNF 中的 4 个种类及其单个流量请求。 $x$  为实验中生成 VNF 实例的编号除以本次实验中的 VNF 种类，所得结果对应下列网络功能。

表 5.1 其中一次实验中所用 VNF 实例

请求的 VNF 实例：代理，负载均衡，网络地址转换，入侵检测系统	
VNF 实例名称	流量请求
代理 $\leftarrow x \% 4 == 0$	80
负载均衡 $\leftarrow x \% 4 == 1$	84

续表 5.1 其中一次实验中所用 VNF 实例

网络地址转换 $\leftarrow x \% 4 == 2$	70
入侵检测系统 $\leftarrow x \% 4 == 3$	89

表 5.2 展现了第一种情况下，继表 5.1 中 50 个虚拟机的信息，包括处理一个单元所需时间和虚拟机的总容量。

表 5.2 其中一次实验中虚拟机的信息

虚拟机 编号	虚拟机容量 (单位: 个)	处理单元时间 (单位: ms)	虚拟机 编号	虚拟机容量 (单位: 个)	处理单元时间 (单位: ms)
0	2	6	25	2	3
1	3	6	26	2	5
2	1	5	27	3	6
3	3	4	28	1	5
4	2	5	29	3	4
5	1	6	30	1	5
6	3	5	31	1	3
7	3	4	32	3	3
8	3	4	33	3	4
9	3	6	34	3	6
10	3	3	35	3	5
11	2	6	36	3	6
12	3	3	37	2	3
13	3	3	38	3	3
14	3	4	39	3	6
15	3	4	40	2	3
16	3	6	41	3	3
17	3	3	42	3	3
18	3	3	43	3	4
19	2	6	44	3	3
20	3	3	45	1	6
21	3	5	46	3	5
22	1	3	47	3	3
23	3	3	48	3	4
24	1	5	49	3	6

由于网络拓扑图是 50 个虚拟机组成的完全图，共计有 1225 条虚拟链路。同时，100 个 VNF 实例中，每个 VNF 实例的偏好列表中各有 50 个虚拟机结点，每个虚拟机的偏好列表中又各有 100 个 VNF 实例。如在此列出具体数据，略显冗长，故不在此列出，但在 5.3 节会分析该算法的表现以及与其他算法的比较。

### 5.3 仿真结果

就 5.2 节所展现的数据，我们可以得出如下的分布式联盟组合：

表 5.3 就 5.2 节所给数据得出的分布式联盟组合

虚拟机 编号	在时隙 $t$ 时，该虚拟机上处理的虚拟 网络功能实例	虚拟机 编号	在时隙 $t$ 时，该虚拟机上处理的虚拟网 络功能实例
0	IDS 13, Proxy 22	25	IDS 4, IDS 6
1	IDS 2, Proxy 13, IDS 14	26	/
2	Load Balance 15	27	Proxy 4, Proxy 6, Load Balance 23
3	Proxy 10, IDS 19, IDS 23	28	Load Balance 0
4	NAT 9, Proxy 7	29	IDS 17, Proxy 21
5	NAT 7	30	NAT 16
6	IDS 10, IDS 5, Proxy 15	31	IDS 9
7	Proxy 9, IDS 11, NAT 8	32	Load Balance 1, Load Balance 5
8	Load Balance 14, Proxy 23, IDS 15	33	IDS 24, Load Balance 22
9	Load Balance 6, NAT 5, NAT 13	34	NAT 23, NAT 22, IDS 7
10	NAT 10	35	Load Balance 3, Load Balance 10, Load Balance 20
11	Proxy 12	36	Proxy 0
12	IDS 8, IDS 18	37	Load Balance 12, Load Balance 11
13	Load Balance 8, Proxy 5, Proxy 11	38	NAT 1, NAT 14
14	Load Balance 16, NAT 15, Load Balance 21	39	Load Balance 7, Load Balance 9
15	Proxy 20, Proxy 8, IDS 1	40	IDS 16, IDS 20
16	NAT 17, NAT 11, Load Balance 13	41	NAT 0, Proxy 24, NAT 2
17	NAT 21, IDS 22, NAT 20	42	IDS 3
18	Load Balance 19, Load Balance 4, Load Balance 2	43	NAT 3, IDS 0

续表 5.3 就 5.2 节所给数据得出的分布式联盟组合

19	/	44	/
20	Load Balance 17, IDS 12, Load Balance 24	45	/
21	Proxy 19, Proxy 18	46	Load Balance 18, NAT 19, Proxy 2
22	NAT 18	47	Proxy 3, Proxy 16, NAT 12
23	/	48	NAT 4, NAT 24, Proxy 14
24	Proxy 1	49	NAT 6, Proxy 17, IDS 21

由 4.3 可知, 经过“分布式联盟算法”后得到的虚拟机与 VNF 实例组成的联盟一定稳定, 不存在任何阻塞组, 即 VNF 实例可以在该联盟中的虚拟机上得到最好的执行效果。图 5.2 展现了 30, 50 和 100 个虚拟机结点全联通网络拓扑图中处理 100 条 VNF 服务链十次实验 VNF 调度的完成时间。在 50 个虚拟机的拓扑图中, 我们可以看到在第 4 次与第 10 次仿真时调度时间明显减少, 这是因为每次从 5.2 章所提及的五个网络功能中随机选取其中的 1-5 个网络功能, 在这两次仿真中, 恰巧只选取了 2 个网络功能, 导致调度时间与其他仿真相比大幅度减少。由于每个 VNF 实例在虚拟机上处理的时间为 10-30ms, 加上对应的传输转发时间, 所以理论上该实验中 VNF 调度总时间为 1000-4500ms 之间, 所以该两次仿真仍然符合实验要求。(同理适用于 30 个结点的第 7 次仿真与 100 个虚拟机的拓扑图中的 2, 3, 5 次仿真)

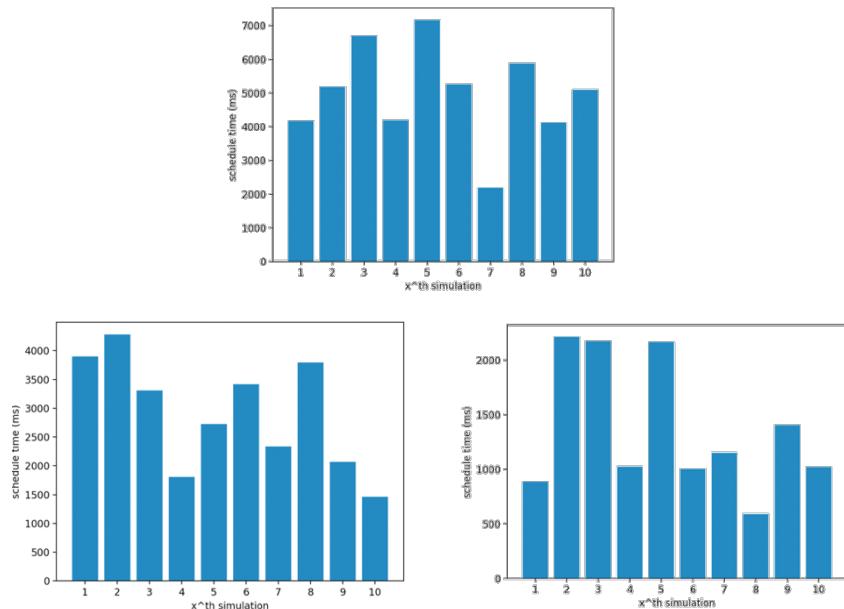


图 5.2 从左到右依次为 30, 50 和 100 个虚拟机结点的网络拓扑图中  
处理 100 条 VNF 服务链的十次实验 VNF 调度的完成时间

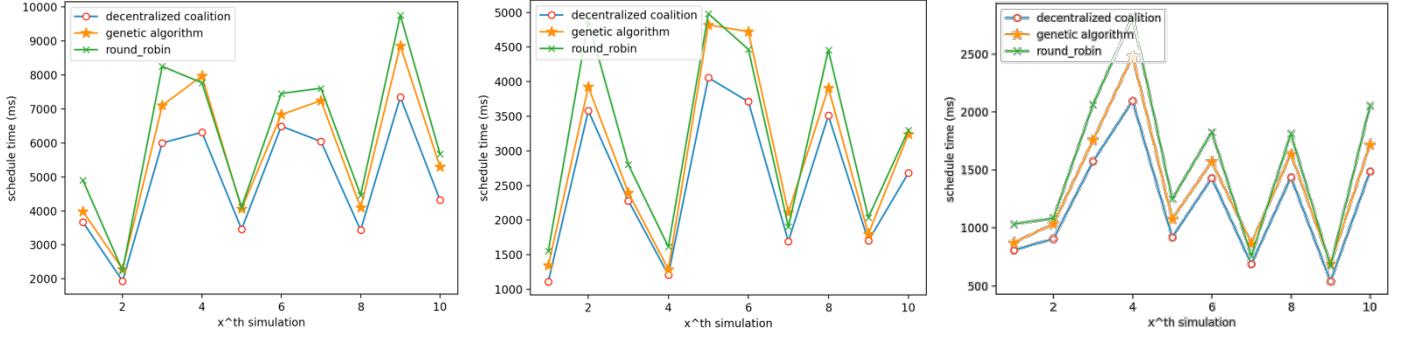


图 5.3 从左到右依次为 30, 50 和 100 个虚拟机结点的网络拓扑图中处理 100 条 VNF 服务链的十次实验 VNF 调度的完成时间的对比图

接着，我们又将该算法与基因遗传算法<sup>[50]</sup>和轮训调度<sup>[30]</sup>算法进行了比较。图 5.3 展现了这三种算法在上述含有 30, 50, 100 个虚拟机结点的全联通网络拓扑图中处理 100 条 VNF 服务链十次仿真中的对比表现。与基因遗传算法与轮训调度算法相比，我们设计的“分布式联盟算法”具有更好的表现。需要注意的是，由于图 5.2 与图 5.3 分别进行了两次不同的仿真（否则柱形图将于折线图有重合），所以对应的时间不同。

同时，我们也进行了个含有 50 个虚拟机结点全联通网络拓扑图用以处理 10 条，20 条，30 条，40 条和 50 条 VNF 服务链的仿真。使用“分布式联盟算法”的完成时间如图 5.4 所示，与遗传基因和轮训调度算法的时间比较如图 5.5 所示。

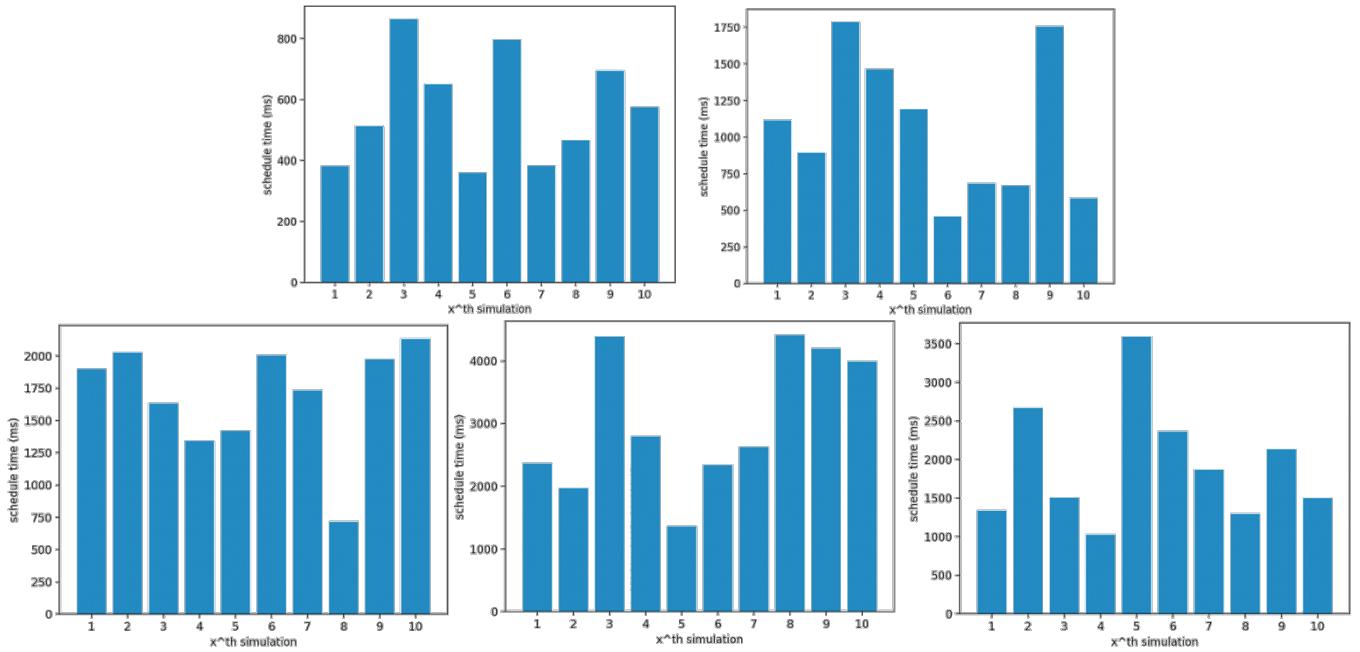


图 5.4 从左上到中依次为处理 10, 20, 30, 40, 50 条 VNF 服务链的完成时间

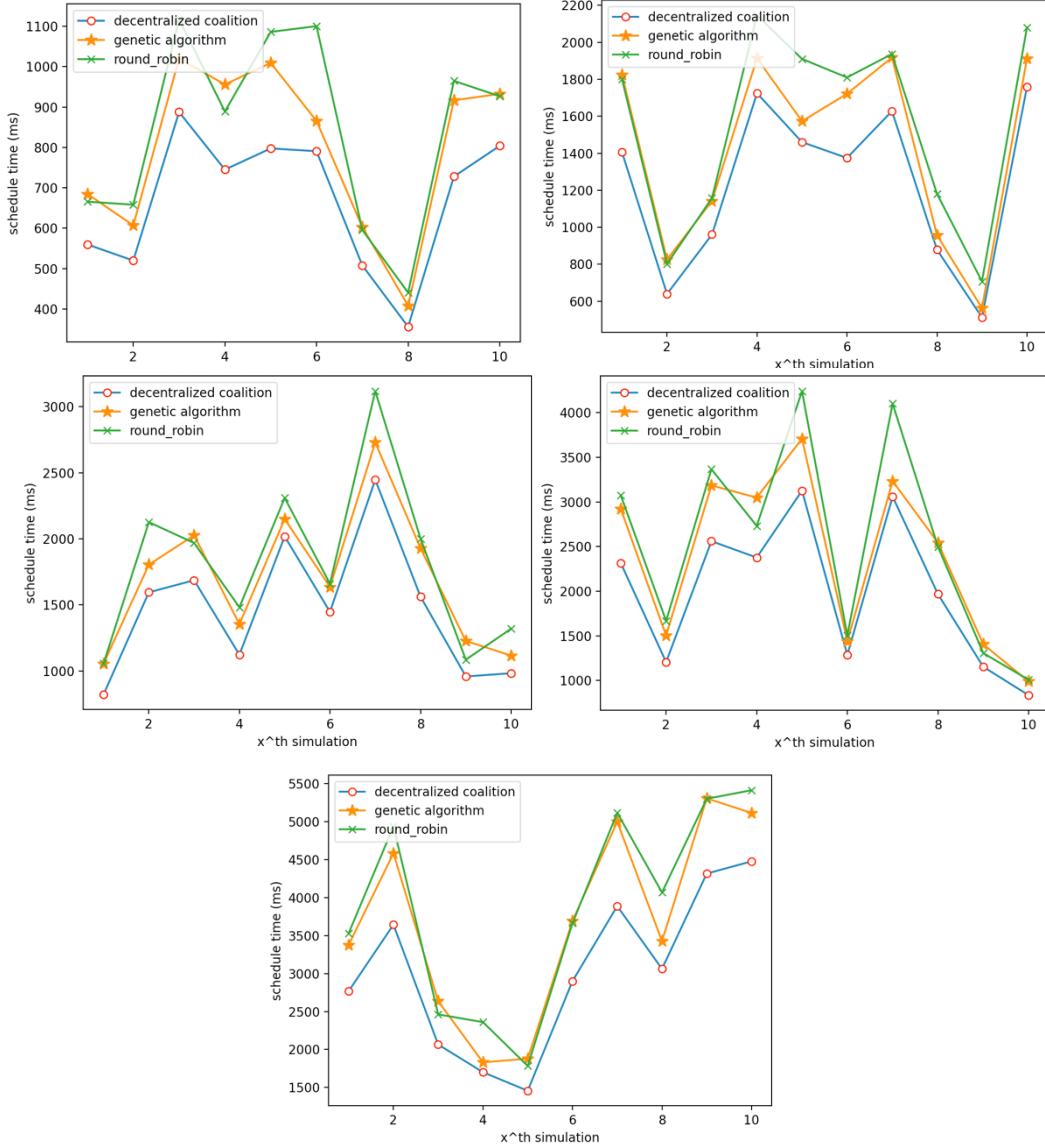


图 5.5 从左上到中依次为三种算法处理  
10, 20, 30, 40, 50 条 VNF 服务链的完成时间的对比图

## 5.4 分析与讨论

从上述实验结果来看, 我们发现基于 Gale-Shapley 算法而扩展的“分布式联盟算法”确实具有更好的表现, 而且可以得到一个稳定的联盟结构, 存在着纳什均衡。

但是还有一些问题值得我们进一步思考。一是, 因为本文所设计的模型与算法都是离线的, 即不可以根据实际情况来灵活调整。比如说在进行 VNF 调度的时候, 可能整个网络拓扑图会发生改变, 虚拟机出现宕机, VNF 实例发生故障, 同时也有可能出现虚

拟机或者 VNF 实例数量，性能，流量请求等发生改变。这时候，无论是虚拟机结点的偏好列表或者是 VNF 实例的偏好列表都需要重新计算，而该算法却不能很好的适应这些改变。在第 2 章文献综述中提及了一些学者使用在线学习算法（online-learning algorithm）或者多臂老虎自动机（multi-armed bandit algorithm）来进行动态实时地部署 VNF 实例，我们觉得该想法也可以运用到 VNF 调度当中，理应得出更佳的结果。

二是，本文将博弈论的思想运用到了 VNF 调度问题中，但是在 VNF 部署中，该思想与稳定婚姻问题应该也具有可以运用之地。基于 Gale-Shapley 的“分布式联盟算法”的本质是参与博弈的用户共享某一有限的资源，从而实现资源利用的最大化以及自身从该联盟中获益的最大化。该思想与 VNF 部署问题中如何在给定的网络资源中部署 VNF 实例从而最小化整体的资源成本相吻合。

上述两个方向可以为下一阶段的研究做铺垫。

## 5.5 本章小结

本章对第 3 章定义的模型和第 4 章中设计的算法进行仿真，并展现了仿真结果（具体给出其中一次的调度结果与十次随机仿真的总结果）。本章的测试结果说明了本文所带来的创新性贡献，可以观察到“分布式联盟算法”相比于基因遗传算法与轮训调度算法，确实具有更少的调度完成时间。由于仅与两种算法进行比较，暂无法确定是否为最优算法，但由于纳什均衡的存在，该算法得到了一个稳定联盟结构，即无论是虚拟机还是 VNF 实例，都对彼此感到“满意”。

## 结 论

随着互联网技术的发展，云计算在人类生活中扮演者越来越重要的角色。从最初的仅仅是云存储（例如百度云网盘）发展为现在的云计算支持的 APP（例如阿里与腾讯旗下的众多移动端应用）。在未来的发展中，云计算将会持续发展为人类赖以生存的技术。但是正如本文第 1 章所言，云计算带来的一系列挑战也成为人们研究的领域，从而诞生出软件定义网络，网络功能虚拟化等工业界和学术界关注的焦点。而本文正是基于边缘云网络，目的是提高虚拟网络功能实例调度的完成时间与提高虚拟机与虚拟网络功能实例所组成联盟的稳定性，从而提升边缘网络的计算能力，本文的贡献如下：

首先，本文通过阅读大量文献，留下较多笔墨来阐述自 2012 年 10 月欧洲电信标准协会的规范小组出版了白皮书以来，学术界针对虚拟网络功能，这一新型的研究领域所做出的贡献。目前，学术界更多地关注虚拟网络功能的部署与动态网络功能链这一方向，却在虚拟网络功能服务链的调度方向上缺少足够数量的研究。本文图文并茂地向读者展现了上述两个方向上较为常用的算法与模型，但也提出了部分文章欠缺思考的部分，该部分为本文之后模型与算法的提出做出了铺垫。

其次，根据上述文献的阅读，本文提出了适用于本文算法的模型。在虚拟网络功能调度问题上，本文考虑了之前很多研究未考虑到的在虚拟机之间相互传输所用的传输时延，以及在同一时隙时，虚拟机可以处理多个虚拟网络功能实例。上述两点其他文献中忽略的方面，提出了较为完整的模型。该问题是一个 NP-hard 问题，并不能在多项式时间内被解决。

再次，我们设计了“分布式联盟算法”用于解决第 3 章定义的模型与问题。该算法基于 Gale-Shapley 算法，但与其不同的是，这不再是一个一对一的匹配问题，而是一对多的匹配算法。我们分别定义了虚拟机与虚拟网络功能实例的偏好列表，并通过三段伪代码阐明了算法的实现过程。我们又给出了一个简单案例，方便读者的理解。同时，我们也证明了该算法没有丢失 Gale-Shapley 算法的稳定性，并存在纳什均衡。

最后，本文根据上述的模型，问题定义与算法设计，进行了实验仿真。实验结果证明，该算法具有不错的调度完成时间，同时与之前学者使用基因遗传算法与轮训调度算法来解决该问题相比，该算法可以提升匹配的性能。

当然，本文仍然存有不足之处。如本文 5.4 节所言，分布式联盟算法暂时无法动态地调度虚拟网络功能实例，一旦在执行过程中，网络发生改变，该算法的应对能力较弱。同时该算法有能力应用在虚拟网络功能部署的问题中，由于时间与本文篇幅有限，该方面的研究可以在未来的工作中实现。

## 参 考 文 献

- [1] Forecast G M D T. Cisco visual networking index: Global mobile data traffic forecast update, 2017 - 2022[J]. Update, 2019.
- [2] Infrastructure report 2014: Ofcom's second full analysis of the UK's communications infrastructure.
- [3] Lu P, Sun Q, Wu K, et al. Distributed online hybrid cloud management for profit-driven multimedia cloud computing[J]. IEEE Transactions on Multimedia, 2015, 17(8): 1297–1308.
- [4] Xue N, Chen X, Gong L, et al. Demonstration of OpenFlow-controlled network orchestration for adaptive SVC video manycast[J]. IEEE Transactions on Multimedia, 2015, 17(9): 1617–1629.
- [5] Yao J, Lu P, Gong L, et al. On fast and coordinated data backup in geo-distributed optical inter-datacenter networks[J]. Journal of Lightwave Technology, 2015, 33(14): 3005–3015.
- [6] Zhu Z, Kong B, Yin J, et al. Build to tenants' requirements: On-demand application-driven vSD-EON slicing[J]. IEEE/OSA Journal of Optical Communications and Networking, 2018, 10(2): A206–A215.
- [7] Alliance N. 5G white paper[J]. Next generation mobile networks, white paper, 2015, 1.
- [8] Greenberg A, Hamilton J, Maltz D A, et al. The cost of a cloud: research problems in data center networks[J]. 2008: 68–73.
- [9] Cuervo E, Balasubramanian A, Cho D, et al. MAUI: making smartphones last longer with code offload[C]//Proceedings of the 8th international conference on Mobile systems, applications, and services. San Francisco, California, United States, 2010: 49–62.
- [10] Satyanarayanan M, Bahl P, Caceres R, et al. The case for vm-based cloudlets in mobile computing[J]. IEEE pervasive Computing, 2009, 8(4): 14–23.
- [11] Bonomi F, Milito R, Zhu J, et al. Fog computing and its role in the internet of things[C]//Proceedings of the first edition of the MCC workshop on Mobile cloud computing. New York, United States, 2012: 13–16.
- [12] Nunes B A A, Mendonca M, Nguyen X N, et al. A survey of software-defined networking: Past, present, and future of programmable networks[J]. IEEE Communications Surveys & Tutorials, 2014, 16(3): 1617–1634.
- [13] Xia W, Wen Y, Foh C H, et al. A survey on software-defined networking[J]. IEEE Communications Surveys & Tutorials, 2014, 17(1): 27–51.

- [14] Jain R, Paul S. Network virtualization and software defined networking for cloud computing: a survey[J]. IEEE Communications Magazine, 2013, 51(11): 24–31.
- [15] Fundation O N. Software-defined networking: The new norm for networks[J]. ONF White Paper, 2012, 2: 2–6.
- [16] Hawilo H, Shami A, Mirahmadi M, et al. NFV: state of the art, challenges, and implementation in next generation mobile networks (vEPC) [J]. IEEE Network, 2014, 28(6): 18–26.
- [17] Drutskoy D, Keller E, Rexford J. Scalable network virtualization in software-defined networks[J]. IEEE Internet Computing, 2012, 17(2): 20–27.
- [18] Nash J F. Equilibrium points in n-person games[J]. Proceedings of the national academy of sciences, 1950, 36(1): 48–49.
- [19] Shoham Y. Computer science and game theory[J]. Communications of the ACM, 2008, 51(8): 74–79.
- [20] Hu J, Wellman M P. Nash Q-learning for general-sum stochastic games[J]. Journal of machine learning research, 2003, 4: 1039–1069.
- [21] Manshaei M H, Zhu Q, Alpcan T, et al. Game theory meets network security and privacy[J]. ACM Computing Surveys (CSUR), 2013, 45(3): 1–39.
- [22] Wang T, Sun Q, Ji Z, et al. Multi-layer graph constraints for interactive image segmentation via game theory[J]. Pattern Recognition, 2016, 55: 28–44.
- [23] Quijano N, Ocampo-Martinez C, Barreiro-Gomez J, et al. The role of population games and evolutionary dynamics in distributed control systems: The advantages of evolutionary game theory[J]. IEEE Control Systems Magazine, 2017, 37(1): 70–97.
- [24] Stallings W. Foundations of modern networking: SDN, NFV, QoE, IoT, and Cloud[M]. Addison-Wesley Professional, 2015.
- [25] Li Y, Chen M. Software-defined network function virtualization: A survey[J]. IEEE Access, 2015, 3: 2542–2553.
- [26] Basta A, Kellerer W, Hoffmann M, et al. Applying NFV and SDN to LTE mobile core gateways, the functions placement problem[C]//Proceedings of the 4th workshop on All things cellular: operations, applications, & challenges. 2014: 33–38.
- [27] Paul S, Jain R. Openadn: Mobile apps on global clouds using openflow and software defined networking[C]//2012 IEEE Globecom Workshops. IEEE, Anaheim, CA, USA, 2012: 719–723.
- [28] Jin X, Li L E, Vanbever L, et al. Softcell: Scalable and flexible cellular core network architecture[C]//Proceedings of the ninth ACM conference on Emerging networking experiments and technologies, New York, United States, 2013: 163–174.

- [29] Erran L, Morley L Z, Rexford M J. CellSDN: software-defined cellular networks[J]. 2012.
- [30] Riera J F, Escalona E, Batalle J, et al. Virtual network function scheduling: Concept and challenges[C]//2014 International Conference on Smart Communications in Network Technologies (SaCoNeT). Vilanova I La Geltru, Spain, 2014: 1–5.
- [31] Sekar V, Egi N, Ratnasamy S, et al. Design and implementation of a consolidated middlebox architecture[C]//Presented as part of the 9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12). San Jose, CA, USA, 2012: 323–336.
- [32] Savi M, Tornatore M, Verticale G. Impact of processing costs on service chain placement in network functions virtualization[C]//2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN). IEEE, San Francisco, USA, 2015: 191–197.
- [33] Bouet M, Leguay J, Combe T, et al. Cost - based placement of vDPI functions in NFV infrastructures[J]. International Journal of Network Management, 2015, 25(6) : 490–506.
- [34] Gale D, Shapley L S. College admissions and the stability of marriage[J]. The American Mathematical Monthly, 1962, 69(1): 9–15.
- [35] Chiosi M, Clarke D, Willis P, et al. Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action[C]//SDN and OpenFlow world congress. Darmstadt, Germany, 2012, 48: 1–16.
- [36] Quinn P, Nadeau T. Problem statement for service function chaining[C]//RFC 7498. RFC Editor, 2015.
- [37] Surendra M T, Majee S, Captari C, et al. Service function chaining use cases in data centers[J]. Working Draft, IETF Secretariat, Internet-Draft draft-ietf-sfc-dc-use-cases-06, 2017.
- [38] Martins J, Ahmed M, Raiciu C, et al. ClickOS and the art of network function virtualization[C]//11th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 14). Seattle, WA, USA, 2014: 459–473.
- [39] Hwang J, Ramakrishnan K K, Wood T. NetVM: High performance and flexible networking using virtualization on commodity platforms[J]. IEEE Transactions on Network and Service Management, 2015, 12(1): 34–47.
- [40] Sekar V, Egi N, Ratnasamy S, et al. Design and implementation of a consolidated middlebox architecture[C]//Presented as part of the 9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12). San Jose, CA, USA, 2012: 323–336.

- [41] Jia Y, Wu C, Li Z, et al. Online scaling of NFV service chains across geo-distributed datacenters[J]. IEEE/ACM Transactions on Networking, 2018, 26(2): 699–710.
- [42] Wang X, Wu C, Le F, et al. Online VNF scaling in datacenters[C]//2016 IEEE 9th International Conference on Cloud Computing (CLOUD). IEEE, San Francisco, CA, USA, 2016: 140–147.
- [43] Cao L, Sharma P, Fahmy S, et al. {ENVI}: Elastic resource flexing for Network function Virtualization[C]//9th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 17). Santa Clara, CA, USA, 2017.
- [44] Luizelli M C, Bays L R, Buriol L S, et al. Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions[C]//2015 IFIP/IEEE International Symposium on Integrated Network Management (IM). IEEE, Ottawa, Canada, 2015: 98–106.
- [45] McGrath M J, Riccobene V, Petralia G, et al. Performant deployment of a virtualised network functions in a data center environment using resource aware scheduling[C]//2015 IFIP/IEEE International Symposium on Integrated Network Management (IM). IEEE, Ottawa, Canada, 2015: 1131–1132.
- [46] Clayman S, Maini E, Galis A, et al. The dynamic placement of virtual network functions[C]//2014 IEEE network operations and management symposium (NOMS). IEEE, Krakow, Poland, 2014: 1–9.
- [47] Leivadeas A, Kesidis G, Falkner M, et al. A graph partitioning game theoretical approach for the VNF service chaining problem[J]. IEEE Transactions on Network and Service Management, 2017, 14(4): 890–903.
- [48] Chen X, Zhu Z, Guo J, et al. Leveraging mixed-strategy gaming to realize incentive-driven VNF service chain provisioning in broker-based elastic optical inter-datacenter networks[J]. IEEE/OSA Journal of Optical Communications and Networking, 2018, 10(2): A232–A240.
- [49] Ma N, Zhang J, Huang T. A model based on genetic algorithm for service chain resource allocation in NFV[C]//2017 3rd IEEE International Conference on Computer and Communications (ICCC). IEEE, Chengdu, China, 2017: 607–611.
- [50] Qu L, Assi C, Shaban K. Delay-aware scheduling and resource optimization with network function virtualization[J]. IEEE Transactions on Communications, 2016, 64(9): 3746–3758.
- [51] Pham C, Tran N H, Hong C S. Virtual network function scheduling: A matching game approach[J]. IEEE Communications Letters, 2017, 22(1): 69–72.

- [52] Xu Z, Liang W, Galis A, et al. Throughput maximization and resource optimization in NFV-enabled networks[C]//2017 IEEE International Conference on Communications (ICC). IEEE, Paris, France, 2017: 1-7.
- [53] Oechsner S, Ripke A. Flexible support of VNF placement functions in OpenStack[C]//Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft). IEEE, London, United Kingdom, 2015: 1-6.

## 修改记录

### (1) 毕业设计（论文）题目修改

#### 第一次修改记录:

原题目：基于博弈论的边缘网络虚拟功能管理

修稿后题目：基于博弈论的边缘网络虚拟功能调度研究

### (2) 毕业设计（论文）内容重要修改记录

#### 第二次修改记录:

第 22 页表 4.1，修改前：没有提供虚拟网络功能实例偏好列表的伪代码

修改后：加入虚拟网络功能实例偏好列表的伪代码

#### 第三次修改记录:

第 22 页表 4.2，修改前：没有提供虚拟机实例偏好列表的伪代码

修改后：加入虚拟机实例偏好列表的伪代码

#### 第四次修改记录:

第 23 页表 4.3，修改前：表 4.3 中原算法为 Algorithm 1 Decentralized Coalition Algorithm

修改后：由于加入两段伪代码，现表 4.3 中算法为 Algorithm 3 Decentralized Coalition Algorithm

#### 第五次修改记录:

第 22 页表 4.1，修改前：表 4.1 中原算法为 Algorithm 1 (英文)

修改后：表 4.1 中原算法为 Algorithm 1 (中文)

#### 第六次修改记录:

第 22 页表 4.2，修改前：表 4.2 中原算法为 Algorithm 2 (英文)

修改后：表 4.2 中原算法为 Algorithm 2 (中文)

#### 第七次修改记录:

第 23 页表 4.3，修改前：表 4.3 中原算法为 Algorithm 3 (英文)

修改后：表 4.3 中原算法为 Algorithm 3 (中文)

#### 第八次修改记录:

第 20 页 3.3.1 章，修改前：3.3 “定理 1：该 VNF 调度问题 (3.1) 是非确定性多项式困难问题 (NP-hard) ” 单独成章

修改后：取消 3.3.1 单独成章，并入到 3.3 章中

#### 第九次修改记录:

第 28 页 5.2 章, 修改前: 只有 50 个虚拟机一种网络拓扑图

修改后: 新增 30 个虚拟机与 100 个虚拟机两种网络拓扑图

**第十次修改记录:**

第 32 页 5.3 章, 修改前: 只有 50 个虚拟机组成的一种网络拓扑实验数据图

修改后: 新增 30 个与 100 个虚拟机两种网络拓扑图的实验数据图与算法对比图

**第十一次修改记录:**

第 32 页 5.3 章, 修改前: 无 10, 20, 30, 40 条服务链的仿真实验

修改后: 新增 10, 20, 30, 40 条服务链的仿真实验

**第十二次修改记录:**

第 33 页 5.3 章, 修改前: 无 10, 20, 30, 40 条服务链的三种算法对比的仿真实验

修改后: 新增 10, 20, 30, 40 条服务链的三种算法对比的仿真实验

**第十三次修改记录:**

第 21 页 4 章, 修改前: 对于虚拟网络功能实例调度的分散联盟的形成

修改后: 分布式联盟构建机制

(3) 毕业设计(论文)正式检测重复比

正式检测重复比为 2%。

记录人(签字): 王威然

指导教师(签字): 夏秋粉

## 致 谢

初夏将至，时间如流水，瞬间就将人的记忆变得模糊，仿佛昨天我还是那个背上行囊，正准备前往大连的高中毕业生。在写到此部分前，幻想过很多次写到“致谢”时的情形，有很多很多的心里话想在此落笔，但真正写到这部分，却有些无语凝噎，竟不知该写些什么。

回首大学四年，有为了自己的梦想奋斗过，有实现梦想时候的振臂欢呼；也有与自己追求的目标擦肩而过的悲伤与不甘；更有迷茫时，站在篮球场仰望茫茫星空与一个人躲在被窝里面的呆滞。但这四年至少过得问心无愧，也实现了很多很多当初从来不敢想象的目标。但如果再给我一次上大学的四年时光，我想自己可能会做的更好，至少不会再因为自己的懦弱与懒惰错过很多很多摆在眼前的良机。

很感谢大学四年教会我知识的所有老师，更感谢给我进实验室看论文，做研究的徐子川老师。但我最想感谢的却是一堂大学课也没有教过我的樊宇老师。因为自己是软件日强专业的原因，本科四年没有一堂英语课，从来没有上过樊老师的英语课，但樊老师教会的我却是做人的道理，和樊老师相处的两年内，一起去过武汉参加过 15 届中国模拟联合国大会，一起去过美国纽约参加 2019 全美模拟联合国大会。很感谢樊老师能在我难受的时候打电话安慰我，也在我低落的时候给我继续向前走的勇气。本科阶段，有这两位老师足矣。

人类的命运仿佛是被安排好了一般，在 2019 年纽模时，我所在的委员会是联合国工业发展组织，当初我们的议题恰好是“信息通讯技术在工业发展中的应用”。我根据当初在实验室读过的论文，在会场上提出了使用云计算来解决中小型企业与欠发达国家没有足够机会来接触 ICT 的问题。这一提案很受在会场其他国家代表的欢迎，我们也将该条提案写进了最终的决议草案中，最后也被收录到联合国学术影响力，作为联合国以后制定决议草案的重要凭证。虽然那时候的我对于云计算的理解并没有现在深，但我第一次发现了自己的乐趣与之后想学习的方向。而巧合的是，徐老师的研究方向正是云计算，边缘计算，软件定义网络与网络功能虚拟化。因为樊老师，所以去了纽约参会；又因为徐老师，所以可以在会场上提出云计算的方法。所以当我看到大工与 KTH 有 4+2 联合培养方案，而 KTH 也是积极响应联合国 17 个可持续性发展目标时，就毫不犹豫地报了 ICT 创新-云与基础网络设施方向。而最后我也成功被录取，也继续可以在这个方向中走的更远，与这篇毕设相似的是，我希望未来有一天可以将我 19 年在决议草案中写的条款变成现实，虽然那个时候提出的见解较为粗糙，但是却足够具有诱人。也希望以后有机会可以去国际电信联盟或者空客去工作。唯一可惜的是，本科阶段还是没能跟

徐老师一起发表一个论文，但受徐老师和之前在纽约参会的影响，我研究生阶段的研究目标跟徐老师相似，以后一定有机会的吧。

最后还想感谢大学阶段陪伴我的同学，舍友，模联协会的小伙伴。最想感谢的是来自南京大学的陆柯君同学，在我申请季最迷茫的时候的鼓励，还有在我生日时写给我的寄语，每当我不知所措的时候就会拿出来看看，真的非常感谢。

大学四年，我做了很多看起来与升学毫无相关的事情，但正是这些事情让我明白了自己究竟喜欢什么，自己未来究竟想做些什么。未来的路肯定会比现在的更为艰辛，去国外和其他更优秀的学生一起竞争肯定会比现在更具有挑战性，但有了本科四年的经验，我想一定可以做的更好。

Le vent se lève, il faut tenter de vivre. 这是我最喜欢的台词，出自宫崎骏的《起风了》。  
现在风正起，人生不可言弃呐。