基于VNF的网络服务冲突检测与避免机制

李政宇[1)] 王兴伟[1)] 易 波[1)] 黄 敏[2)]

[1)](东北大学计算机科学与工程学院 沈阳 110169)

[2)](东北大学信息科学与工程学院 沈阳 110819)

摘 要 针对如何构建服务功能链(Service Function Chain,SFC)并进行资源分配为用户提供满意服务的问题,本文提出了一种包括候选路径构建、依赖冲突检测与避免以及虚拟网络功能(Virtual Network Function,VNF)部署的机制.首先,为了给SFC部署提供充足的资源,提出二级筛选及最优化选取的候选路径构建规则,为服务提供预选路径.其次,在SFC构建过程中,检测复用性与依赖关系之间的冲突,将依赖关系划分二元组后进行冲突判断,若产生冲突则进行等价类划分,给出冲突集合.然后,提出基于冲突集合以及LFGL(Least-First-Greatest-Last)原则的VNF部署规则,以最大化链路剩余带宽,保证端到端延迟.最后,在进行服务递交时,检测VNF流入流出比对数据量的影响,若产生冲突则进行冲突避免,若无法成功避免则执行规避策略.最后基于小型和大型两种网络拓扑对仿真系统进行性能评价.实验结果表明,本文设计的机制在复用率、时延、部署成功率方面所表现出的性能均优于对比算法.

关键词 NFV;VNF;SFC;冲突与避免

中图法分类号 TP393 **DOI号** 10.11897/SP.J.1016.2023.00385

Network Service Conflict Detection and Avoidance Mechanism Based on VNF

LI Zheng-Yu[1)] WANG Xing-Wei[1)] YI Bo[1)] HUANG Min[2)]

[1)](*College of Computer Science and Engineering, Northeastern University, Shenyang* 110169)

[2)](*School of Information Science and Engineering, Northeastern University, Shenyang* 110819)

Abstract Aiming to address the problem of how to build a service function chain (SFC) and allocate resources to provide users with satisfactory services, a mechanism including candidate path construction, dependency conflict detection and avoidance, and virtual network function (VNF) deployment is proposed. First, in order to provide sufficient resources for SFC deployment, the candidate path construction rules of second-level filtering and optimal selection are proposed to provide pre-selected paths for services. Secondly, during the construction stage of SFC, the conflicts between reusability and dependency are detected by the mechanism, the dependencies of the resources are divided into binaries. By means of the division, the conflicts can be judged afterwards. Equivalence class division is performed if conflicts arise, following by the provision of the set of conflicts. Then, a VNF deployment rule based on conflict set and LFGL (Least-First-Greatest-Last) principle is proposed. The rule maximizes the remaining link bandwidth and guarantees end-to-end delay. Finally, when the service provider submits the required service, the impact of VNF inflow-outflow ratio on data volume is detected, conflicts are avoided if they arise, and evasion strategies are executed when comes to the facts that the conflicts cannot be avoided successfully. Finally, the performance of the simulation system is evaluated based on small and large network

收稿日期:2021-11-15;在线发布日期:2022-09-30.本课题得到国家重点研发计划项目(2019YFB1802800)和国家自然科学基金项目(62002055, 62032013, 61872073)资助.**李政宇**,硕士研究生,主要研究方向为软件定义网络、流量工程、人工智能.E-mail:1666854587@qq.com.**王兴伟**(通信作者),博士,教授,博士生导师,中国计算机学会(CCF)高级会员,主要研究领域为未来互联网、云计算、网络安全和信息安全等.E-mail:wangxw@mail.neu.edu.cn.**黄 敏**,博士,教授,博士生导师,主要研究领域为智能算法设计与优化、调度理论与方法等.**易 波**,副教授,主要研究方向为网络功能虚拟化和服务链等.

topologies. The experimental results show that the mechanism designed in this paper performs better than the comparison algorithm in terms of reuse rate, delay and successful deployment rate.

Keywords network function virtualization; virtual network function; service function chain; conflicts and avoidance

1 引 言

传统网络架构下,服务供应商需要在网络中部署大量的专有设备(如中间件)以满足用户需求,中间件的部署导致了高额的运营支出与成本支出,也使传统网络变得臃肿和僵化[1]. 网络功能虚拟化(Network Functions Virtualization, NFV)以软件的形式代替现有网络功能,该软件被称作虚拟网络功能(Virtual Network Function, VNF). NFV 架构实现了软硬件解耦,在降低支出的同时,使网络功能更新变得更加灵活. 然而,NFV 在提升网络灵活性的同时,也带来了新的挑战,例如:如何构建服务功能链(Service Function Chaining, SFC)并进行资源分配,为用户提供满意的网络服务.

随着网络服务功能的不断激增,人们对网络服务提出了更高的要求. NFV 架构的出现一定程度上解决了该问题,有效地为服务提供者降低了成本开销,缩短了部署周期,同时也增加了网络的灵活性. 尽管各领域一直在努力使 NFV 成为现实,但仍存在一些未解决的问题,例如:VNF 生命周期管理与编排. 提高资源利用率、降低 VNFs 部署成本和开销成本也是一个备受关注的问题. 由于 VNFs 存在生命周期,当网络中存在大量已部署的 VNFs 时,对于新到达的请求,可以根据已部署 VNFs 的剩余资源、剩余生命周期以及请求包含的网络功能等情况进行综合判断,为其构建复用率高的路径,以提高当前网络中已部署的 VNFs 的复用率,从而可有效提高资源利用率. 同时,在网络处理能力不变的情况下,亦可有效提高网络吞吐量,并且能降低重复部署 VNF 的部署成本和运行相同 VNFs 所造成的开销成本. 而在构建复用率较高的路径的过程中,因为服务请求中包含 VNFs 的依赖关系,且已部署的 VNFs 存在物理序列限制,所以很可能出现 VNFs 依赖冲突.

为有效提高 VNFs 复用率,降低 VNFs 部署成本和开销成本,高效利用网络资源并提高网络吞吐量,本文对基于 VNF 的网络服务进行冲突检测与避免. 当网络服务请求到达后,通过一系列限制条件,为其构建一条复用率最大、代价最小的候选路径,根据请求中的 VNF 依赖关系对路径上已部署的 VNF 进行调整,在保证复用率最大的同时满足请求的依赖关系. 由于经过 VNF 的顺序不同会导致数据量发生改变,所以完成 VNF 部署顺序调整后,需要对路径是否满足带宽限制进行判断,当候选路径不能满足当前服务请求时,需要对路径进行动态调整.

本文的主要贡献如下:

(1)综合考虑 VNF 复用性、时延、跳数等因素,提出了候选路径的构建方法.

(2)在依赖冲突检测与避免模块,提出了一种冲突划分方法及针对不同冲突的解决方案.

2 相关工作

NFV 是一个杰出的网络体系结构概念,它改变了对网络功能的管理. 其能否被广泛采用主要取决于能否确保有效地进行资源分配,以防止资源过剩或不足. 因此,在保证可接受的端到端延迟的同时,以最具有成本效益的方式放置网络功能非常重要. 目前,SFC 构建以及 VNF 部署中存在一些特征问题,很多研究者在这两个领域开展了大量研究.

2.1 SFC 部署与 VNF 部署

SFC 部署问题的研究主要分为两类:一类是针对单条 SFC;一类是针对多条 SFC. 在文献[2]中主要研究了多条 SFC 之间的映射关系,其目标是最大程度降低网络资源消耗. 提出一种基于列生成模型和最短路径流量分组的启发式两阶段模型,在相对少的时间里解决复杂问题. 文献[3-6]研究单条 SFC 的构建或者部署关系,在构建路径时,综合考虑了最大化链路剩余带宽,最小化使用节点个数以及最小化时延. 在文献[3-6]中,主要对 VNF 服务功能链的部署问题进行研究,其 VNF 部署顺序与 SFC 顺序一致. 文献[3]主要目标为降低部署 VNF 的总成本. 文献[4]为提高 VNF 利用率、减少链路消耗,在部署时考虑了已实例的 VNF 的复用性. 文献[5]在部署时忽略节点能力、延迟以及 VNF 之间的优先级

限制. 文献[6]提出一种优化的整数线性规划模型结合启发式算法，一定程度解决了混合整数线性算法(Mixed Integer Linear Programming，MILP)的复杂度高和耗时的缺点，但并未考虑构建与部署时的特征问题.

2.2　服务链构建与部署相结合

在 VNF 部署问题上，多将其构建为 NP 难问题，通过整数线性模型或者启发式算法来解决相应问题. 在部分研究中，学者亦将服务链构建与部署问题相结合，以更好的分配资源，为租户提供更好的服务. 文献[7]为解决如何合理编排 VNFs 并在物理设备上分配资源，提出一个启发式 CoordVNFs 算法，以协作方式解决服务链构建及其部署问题. 该算法在多项式时间内计算结果，忽略对网络性能的影响，最小化带宽使用率. 文献[8]指出链路构建和服务器使用率之间的关系在功能链部署问题上起到重要作用. 受到压力测试的启发，作者首先根据需求的网络状态和属性弹性，为每个需求动态调整链路和服务器的使用. 然后通过计算合理的路径长度，决定每个 VNF 是否使用额外的服务器资源或重用已存在的服务器资源. 最后，结合链路与服务器的当前状态进行服务链的部署. 文中提出的方法可有效地适应动态网络资源分配. 文献[9]指出，RA(Resource Allocation)问题分为三个阶段，即使整个 NFV-RA 问题存在一个可行解，第一阶段或第二阶段的最优解也会使整个问题的求解陷入局部最优或下一阶段不可行的境地.

综上所述，在 VNF 链路构建与部署中，已有文献的研究更多考虑的是单个阶段部署或各个阶段的联合管理，以实现最优目标，采用的算法多为启发式算法、整数线性模型、混合整数线性模型等方法. 在现有的文献研究中，考虑 VNF 依赖关系以及 VNF 对链路数据量的影响从而进行链路选择和 VNF 部署的研究还不多，考虑网络中已部署的 VNF 的复用和共享因素来完成 VNF 部署和链路构建的文章更少. 因此，本课题基于 VNF 的复用性来研究其链路构建及部署过程，在考虑复用性与共享性的过程中会出现 VNFs 依赖冲突及带宽冲突，为此设计了一种基于 VNF 的网络服务冲突检测与避免的机制.

3　系统分析

3.1　问题描述

在 NFV 架构下基于 VNF 的网络服务请求到达后，VNFs 部署与链路构建中存在的特征问题有如下几点：

(1) VNFs 优先级限制网络功能虚拟化场景的布置；

(2) 带宽需求的改变依赖于流量通过 VNF 实例的顺序；

(3) 生命周期内已经实例化的 VNF 在网内大量存在.

在 NFV 架构中，VNF 依赖关系、VNF 复用性以及 VNF 对链路数据量的影响仍为急需解决的问题，且对其进行研究切实可以提高网络性能. 在现有研究中，考虑 VNF 依赖关系以及 VNF 对链路数据量的影响从而进行链路选择和 VNF 部署的研究还不多，综合考虑网络中已部署的 VNF 的复用和共享因素来完成 VNF 部署和链路构建的文章更少. 若在链路构建和 VNF 部署时，综合考虑以上特征问题，则会产生以下冲突：

(1) 考虑生命周期内已部署 VNF 的复用性，可能会产生物理 VNF 序列与依赖关系冲突；

(2) 基于最大化链路剩余带宽部署 VNF，VNFs 对数据流的影响可能造成已构建路径的带宽冲突.

基于以上特征问题以及可能产生的第一种冲突类型，本文设计基于 VNF 的网络服务冲突检测与避免机制. 该机制将综合考虑三个特征问题来完成服务递交，在服务过程中，若出现上述冲突则通过相应规则进行避免，以完成网络服务. 最终在保证 VNFs 复用率最大的目标下，降低 VNFs 部署成本和开销成本，提高 SFC 的成功部署率、网络资源利用率及网络吞吐量.

3.2　应用场景

如图 1 和图 2 分别展示了不包含依赖关系和含有依赖关系的服务功能请求(Service Function Request，SFR)路径构建及 VNF 部署的问题. 此网络中存在 7 个节点，其中节点 1、节点 4、节点 5、节点 7 与可部署 VNF 的服务器相连接，每个服务器中标明已经部署在网络中的 VNF. 当有新的服务功能请求到达时，需要根据服务请求中的 VNF 种类为服务请求构建新的服务功能链. 当前服务功能请求包含{VNF2，VNF1，VNF7，VNF3}，本部分将使用该请求作为示例来说明依赖关系对于候选路径中 VNF 复用及部署的影响.

图 1 对已部署 VNF 的剩余资源、生命周期和排队时延等进行了综合考虑，为服务请求构建的候选路径如图中曲线所示，该路径上部署的 VNF 包

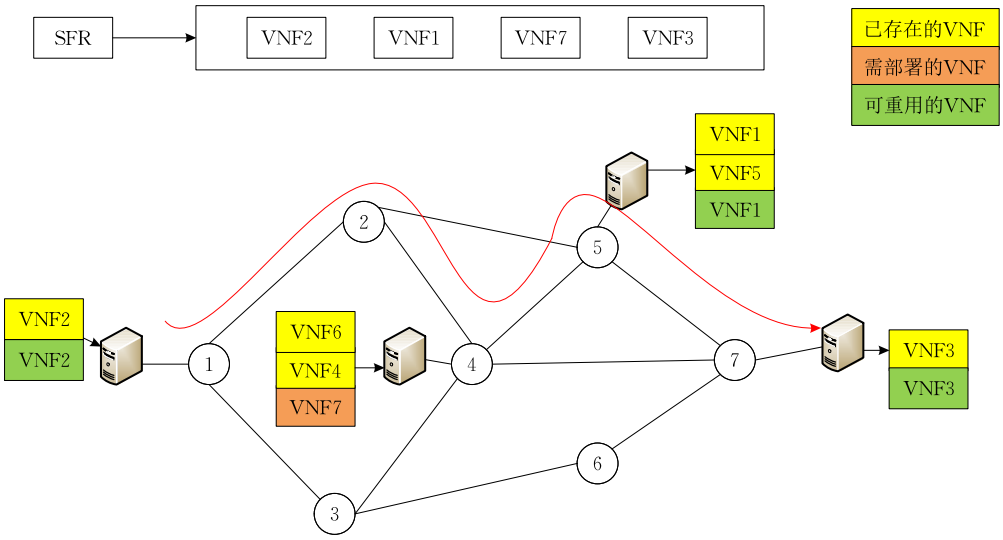

图 1 无依赖关系场景

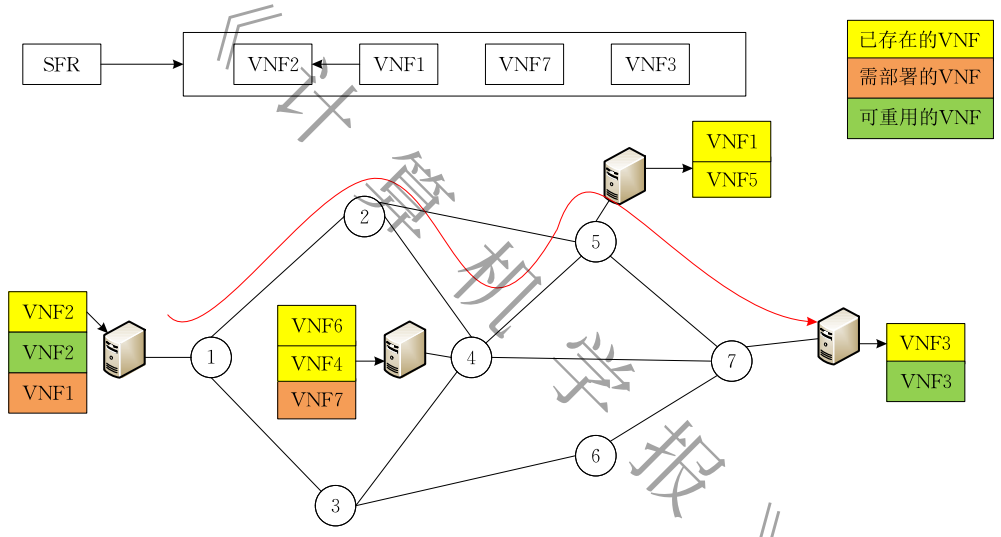

图 2 有依赖关系场景

括{VNF1,VNF2,VNF3,VNF4,VNF5,VNF6},服务请求包含{VNF1,VNF2,VNF3,VNF7}. 在新到来的请求中没有 VNF 的依赖关系,那么对处理请求的虚拟网络功能的顺序就没有特殊的要求. 因此,候选路径中已经存在的 VNF 均可以被复用,即可以复用候选路径中的{VNF1,VNF2,VNF3},仅需要根据一定规则完成 VNF7 的部署,即可满足服务请求.

图 2 展示了有依赖关系的服务请求路径构建及 VNF 部署问题. 已选择的候选路径中已部署的 VNF 有{VNF1,VNF2,VNF3,VNF4,VNF5,VNF6},新到达的服务请求中需要{VNF1,VNF2,VNF3,VNF7}. 由图 2 可知,可复用的 VNF 有{VNF1,VNF2,VNF3}. 但候选路径中可复用 VNF 的物理顺序为{VNF2,VNF1,VNF3},而服务请求中包含了依赖关系{VNF1->VNF2},即数据必须先经过 VNF1 处理,然后再经过 VNF2 处理. 在此情况下,依赖关系的存在导致了可复用 VNF 部署冲突问题. 根据图中部署情况,对节点 1 中的 VNF2 和节点 5 中的 VNF1 进行复用取舍,根据规则选取复用 VNF1 或者 VNF2. 假设复用节点 1 的 VNF2,则需要在节点 1 上重新部署 VNF1,再根据一定的规则完成 VNF7 的部署,以满足服务请求.

所以在构建候选路径时,不仅需要考虑已部署 VNFs 的复用性问题,还需要考虑请求中存在的 VNF 之间的依赖关系.

3.3 模型建立

3.3.1 网络拓扑

在本文中利用无向图 $G(N,L)$ 来描述网络,N 代表网络中节点的集合,L 代表网络中链路的集合.

式(1)描述了节点与虚拟机的连接情况. 其中，$\exists n \in N$，当 $n_{vs}=1$ 时，代表该节点为可部署 VNF 的服务器(VS)节点；当 $n_{vs}=0$ 时，代表该节点为没有部署能力的普通节点.

$$n_{vs}=\begin{cases}1, & n \text{ 存在 } vs \text{ 与之相连接}, n\in N, vs\in VS\\ 0, & n \text{ 不存在 } vs \text{ 与之相连接}, n\in N, vs\in VS\end{cases} \tag{1}$$

3.3.2　虚拟网络功能(VNF)

$VNF(F_j, I_j, sc_j, stor, isfree, packet_{id}, local)$，将 VNF 描述为七元组，其中，$F_j$ 代表第 j 个 VNF 的网络功能，I_j 代表第 j 个 VNF 对网络流量的影响率，sc_j 代表 VNF 所需要的处理能力，$stor$ 代表当前 VNF 所需要的存储资源，$isfree$ 代表所部署的 VNF 是否空闲，$isfree=0$ 表示空闲，$isfree=1$ 表示处理中. $packet_{id}$ 代表当前正在处理的包的标号. $local$ 代表该 VNF 所处的 VS 编号. $Mf=\{F_j \mid j=1,2,\cdots,|Mf|\}$ 为 VNF 集合.

3.3.3　网络服务(SF)

将所有网络服务抽象为集合，如式(2)所示，该集合表示等待被服务的所有请求，包含当前时间点到达的请求和尚未处理的早期请求. 所有请求均需要在当前时刻开始处理.

$$SFset=(NFVR^1, NFVR^2, \cdots, NFVR^i),\ i\geq 0 \tag{2}$$

本文将网络服务请求描述为十元组，$NFVR^i(req\_id, n^i_{init}, n^i_{term}, dr^i_{init}, b_{req}, Mf, R, computer, req\_time, deadline)$，其中，$NFVR^i$ 代表第 i 个服务请求，req_id 代表请求号，n^i_{init} 代表第 i 个服务请求的初始节点，n^i_{term} 代表第 i 个服务请求的目的节点，dr^i_{init} 代表第 i 个服务请求的初始数据量，b_{req} 代表该请求所需最低带宽限制，Mf 代表第 i 个服务请求的 VNF 的集合，R 代表当前请求中 VNFs 集合中的依赖关系，$R=\{F_i \rightarrow F_j\}, i\in\{1,2,\cdots,|Mf|\}$，$|Mf|$ 为请求中所需全部 VNF，该依赖关系代表 VNF_{F_j} 必须在 VNF_{F_i} 之前先执行. $computer$ 代表当前服务所需要的计算资源，req_time 代表到达时间，$deadline$ 代表请求最大可容忍时延.

3.3.4　链路

将链路表示为四元组，节点 u 与节点 v 之间的链路表示 $link(u,v)$，具体定义如式(3)所示：

$$link(u,v)=(s,d,bandwidth_{avi}, conflict_{state}) \tag{3}$$

其中，s 表示当前链路初始节点，d 表示当前链路终点，$bandwidth_{avi}$ 表示当前链路所能提供的带宽大小，$conflict_{state}$ 用于标记当前链路是否发生带宽冲突，$conflict_{state}=0$ 表示未发生冲突，$conflict_{state}=1$ 表示发生冲突.

3.3.5　服务器

当前 VNF 部署场景包含不可分割 VNF、可分割 VNF 和多路径路由. 不可分割 VNF 指一个服务器可包含多个虚拟机，将虚拟机分配给 VNF，一个虚拟机只能部署一个 VNF[10]. 可分割 VNF 是指一个 VNF 的不同组件可被放置到不同的服务器中，每台服务器服务于原始流的子流. 多路由路径是指两个后续 VNF 之间的流可以通过多个路径路由，只适用于可容忍无序交付的网络[11]. 而已有实验表明当一个虚拟机部署多个网络功能时，网络性能会下降[12]. 所以，本文选用不可分割 VNF，即一个服务器可划分为多个 VM，一个 VM 部署一个 VNF.

本文将普通节点定义为二元组，具体表示如式(4)所示：

$$Node=(node_{num}, isconnected\_server) \tag{4}$$

其中，$node_{num}$ 表示节点编号，$isconnected\_server$ 表示当前节点是否与可部署 VNF 的服务器节点相连接，值为 1 代表与服务器相连接，值为 0 代表不与服务器相连接.

本文将服务器构建为一个二元组，具体表示如式(5)所示：

$$Server\_Node=(VNF\_dep, storage) \tag{5}$$

其中，VNF_dep 代表当前服务器已经部署的 VNF 集合，$storage$ 代表当前服务器的总存储资源.

3.3.6　目标函数

目标函数为：在服务请求基础条件限制下最小化 VNF 部署数量，即最大化 VNF 复用率，其公式表示如式(6)所示. 选取这一目标的原因是：部署的 VNF 数量对于网络提供商的成本有重要和直接的影响[4]，提高复用率可有效降低 VNF 部署成本和开销成本. 同时，对于网络性能而言，在 CPU 处理能力相同时，复用率高可增加资源利用率，提高吞吐量. 为达成该目标，对本文的模型进行限制，其具体描述如式(7)～(11)所示.

$$\text{Max}\ \frac{\sum_{j=0}^{|Mf|} VNF^i_{reuse_j}}{|Mf|} \tag{6}$$

$$\text{s.t. } \forall (u,v)\in L, dc(u,v)\leq MaxLinkLoad\leq bc(u,v) \tag{7}$$

$$\sum_{j=1}^{|Mf|} x^i_{F_j}=|Mf|,\ x^i_{F_j}\in(0,1) \tag{8}$$

$$h_j=\text{Min}\Big[0, \sum_{i=1}^{|SFset|} x^i_j\Big],\ j\in NFV_{SFset} \tag{9}$$

$$\sum_{i=1}^{|SFset|}\sum_{j=1}^{|Mf|} \beta^{vs}_{F_{i,j}} sc_j \leq sc(vs),\ \beta^{vs}_{F_{i,j}}\in[0,1] \tag{10}$$

$$q^i_{F_{j'}} < q^i_{F_j}, F_j, F_{j'}\in Mf, F_j\rightarrow F_{j'}, j\neq j', \forall i \tag{11}$$

其中,式(7)表示对带宽进行限制.其中 $bv(u,v)$ 代表 $link(u,v)$ 可获得的带宽,$dc(u,v)$ 代表 $link(u,v)$ 传输的数据量,对于数据流,每一段链路上的数据流量都不超过最大链路负载,并且不应该超过该链路的带宽处理能力.

式(8)对 VNF 部署提出要求,若数据流被处理,则所被处理请求中的 VNF 均需部署完成.其中 $x_{F_j}^i=1$ 代表第 i 个请求中的 VNF_{F_j} 已经部署在服务器上,否则,其值为 0.第 i 个请求中,每个 VNF 仅选择一个服务器(VS)部署.式(9)表示对同一请求中同类 VNF 实例化的个数进行限制.在同一服务请求中,一种 VNF 最多在一个特定服务器上进行一次实例.其中 h_j 表示 j 类型的 VNF,NFV_{SFset} 表示网络服务集合($SFset$)中所有 VNF 的种类.式(10)表示对服务器处理能力进行约束,当前服务器处理能力需求不超过服务器的总处理能力.其中,$\beta_{F_{i,j}}^{vs}=1$ 代表第 i 个请求中,VNF_{F_j} 安装在服务器 vs 上,否则其值为 0.$sc(vs)$ 代表服务器的空间处理能力,对于当前网络中所有的请求,可部署 VNF 的服务器上持有的 VNF 总数不能超过其总处理能力.

式(11)表示对 VNF 处理顺序提出了要求.在含有依赖关系的请求中,服务处理顺序按优先级约束处理.其中 $q_{F_j}^i$ 为 F_j 的执行顺序,$q_{F_{j'}}^i<q_{F_j}^i$ 表示 $F_{j'}$ 必须在 F_j 之后执行.

4 系统架构

4.1 系统模块

系统框架图如图 3 所示,系统主要分为三大模块:候选路径构建、依赖关系冲突检测与避免、VNF 部署.各个模块之间的协调作用如图 3 所示,其中候选路径构建模块主要负责为新到达的请求查找合理路径并进行数据传输以保证最大化 VNF 复用率;依赖冲突检测与避免模块主要负责分析链路已部署 VNF 的物理序列与请求中可复用 VNF 以及请求中的依赖关系是否发生冲突,若发生冲突则进行冲突避免;完成依赖关系冲突检测与避免的处理之后,将不可复用的 VNF 以及产生冲突需要重新部署的 VNF 按照不同原则进行部署,以满足服务请求.

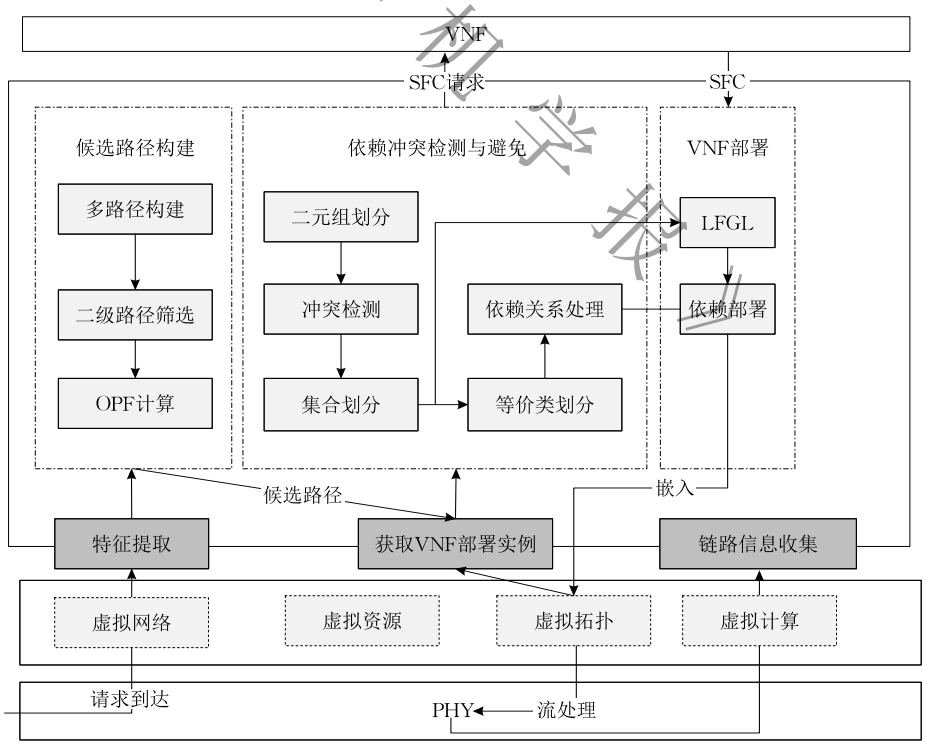

图 3 系统框架

该机制的大致流程如下:

(1)服务请求到达网络,物理节点上传数据包信息至控制器;

(2)控制器对到达的服务请求进行流特征提取,将特征信息提供给候选路径构建模块;

(3)候选路径构建模块根据规则构建最终路径;

(4)下发路径信息至物理层,物理层根据当前链路信息向控制器反馈路径中的可复用 VNF 序列;

(5)检测是否发生依赖关系冲突；

(6)若未发生冲突则在候选路径进行 VNF 部署；若发生依赖冲突，进行冲突避免. 更新相关集合后，进行 VNF 部署.

4.2　候选路径构建机制设计

4.2.1　多路径构建及二级筛选

流特征提取模块提取源节点、目的节点以及请求中的 VNF 集合. 利用 Yen's 最短路径算法计算出 K 条最短路径. 初步舍弃原则如下：

(1)若找到的服务路径中没有节点与可部署 VNF 的服务器相连接，则舍弃该条路径；

(2)如果节点与可部署 VNF 的服务器相连，则进一步判断所有服务器当前剩余处理能力总和能否满足请求中所有 VNF 集合的处理请求，若不能则舍弃该路径.

同一时刻，网络中存在大量服务请求，同一类 VNF 实例数量多则并发程度高，但所有 VNF 实例基于网络中的同一虚拟资源，其总量一定. 某一种类 VNF 所能分配的系统资源计算公式如式(12)所示，$\overline{res_{VNF_k}}$ 代表当前网络继续实例 VNF_k 所能获得的平均虚拟资源，$res_{total}^{VNF_k}$ 代表分配给 VNF_k 的全部资源，$\sum_{i=1}^{num} res_i$ 代表 VNF_k 当前实例所占用的资源总数量，$number$ 代表拟再实例的 VNF_k 的数量.

$$\overline{res_{VNF_k}}=\frac{res_{total}^{VNF_k}-\sum_{i=1}^{num} res_i}{number} \tag{12}$$

若同种 VNF 部署的数量多则其平均获取的资源就少，从而导致每个 VNF 实例的处理能力低，若当前 VNF 处理大量请求，则可能造成网络拥塞. VNF 处理队列将产生上溢，造成网络时延增加甚至请求被大量丢弃. 避免该情况发生除增大已部署 VNF 复用率外，还需要保证网内同种 VNF 间负载均衡. 所以，继续根据候选路径中所需 VNF 的负载进行进一步筛选. 其规则如下：

(1)计算可复用 VNF 处理队列长度 l_i，若队列长度小于处理队列上限 α，即 $l_i<\alpha$，则进行(2)，否则转向(3)继续执行；

(2)代表当前可复用 VNF 队列长度较短，处理时延较小，则保留该路径；取下一条路径转为(1)继续执行，若无可判断路径则结束；

(3)代表当前可复用 VNF 队列长度较长，增加时延甚至可能产生丢包，所以标记导致路径被舍弃的 VNF，将该路径放入后备队列中；取下一条路径继续执行(1)，若无可判断路径则结束.

全部标记完成后，若还有剩余候选路径则执行最优路径选取，选取最终候选路径. 若标记完成后无剩余候选路径，则将后备队列中的路径取出，将该路径标记的 VNF 从可复用集合中剔除，进而执行最优路径选取策略.

4.2.2　最优路径选取

对于舍弃后的 K-n 条路径进行最优化选取，根据式(13)计算出最优因子，选取最优因子最大的路径作为最终的服务候选路径. 并将服务请求中的 VNFs 分别存储到可复用 $VNFsSet_{reuse}$ 与需要部署 $VNFsSet_{deploy}$ 集合.

$$R=\frac{\sum_{j=0}^{|Mf|} VNF_{reuse_j}^{i}}{|Mf|}\Big/L \tag{13}$$

$|Mf|$ 为请求集合中 VNF 的总数量，$VNF_{reuse_j}^{i}$ 代表可重用的 VNF，L 代表链路长度. 式(7)～(11)约束已经保证了当前链路的网络状态较好，而数据流在网络内传输过程中经过的跳越少，对网络整体性能影响越小. 所以尽量选取复用率高且链路节点少的候选路径作为服务的最终路径.

综上所述，根据底层传递的网络服务请求，获取源地址和目的地址，以时延为代价利用 K 最短路径算法计算获取 K 条候选路径. 但选取的 K 条路径考虑因素较少，所以需要进行二级筛选. 候选路径构建的算法的时间复杂度为 $O(n^2)$，算法描述如算法 1 所示.

算法 1.　候选路径构建(CLG).

输入：源节点、目的节点、当前请求所需要的 VNF 集合 $VNF\_needset$、请求允许最大时延

输出：候选路径 $CandidatePath$、复用率、候选路径中已存在 VNF 集合 VNF_{c_road}

```
BEGIN
1.  根据请求所需的 VNF 集合，获取所需的部署能力
2.  利用 Yen's 算法求出 K 条延迟最小路径存入 RoadList、相应最小延迟存入 mindelay
3.  WHILE (i from 0 to RoadList.size()) DO
4.      获取当前路径 i 包含的节点，存入 NodeList
5.      WHILE (j from 0 to NodeList.size()) DO
6.          IF 当前节点与服务器相连，THEN
7.              更新与 VS 相连接的节点 vscount
8.              更新路径总剩余部署能力 vscapacity
9.          END IF
10.     END WHILE
11.     IF vscount==0 OR vscapacity<Deploy_capacity_need THEN
```

```
12.      舍弃 RoadList 中当前路径
13.    END IF
14.    i=i+1, vscount=0, vscapacity=0
15. END WHILE
16. IF RoadList.size()>0,THEN
17.    WHILE (i from 0 to RoadList.size()) DO
18.      获取当前路径所有可复用 VNF 的处理队列长
         度 L 存入 Queue 中
19.      WHILE (j from 0 to Queue.size()) DO
20.        IF VNF 队列长度 L 大于阈值, THEN
21.          将当前 VNF 从可复用集合去除,将当前
             路径转入后备队列 BackupQueue
22.          更新 VNF 可复用集合
23.        END IF
24.      END WHILE
25.    END WHILE
26. ELSE
27.    丢弃该服务请求
28. END IF
29. IF RoadList.size()==0 THEN
30.    WHILE (i from 0 to BackupQueue.size()) DO
31.      获取经二级筛选后,当前路径可复用 VNF 个数
32.      根据式(13)计算最优化因子
33.      IF 当前路径因子小于最优化因子,THEN
34.        更新最优化因子,设置当前路径为候选路径
35.      END IF
36. ELSE
37.    WHILE (i from 0 to RoadList.size()) DO
38.      获取经二级筛选后,当前路径可复用 VNF 个数
39.      根据式(13)计算最优化因子
40.      IF 当前路径因子小于最优化因子,THEN
41.        更新最优化因子,设置当前路径为候选路径
42.      END IF
43.    END WHILE
44. END IF
END
```

4.3 依赖关系冲突检测与避免

获取流特征提取模块返回的当前处理请求中的 VNF 依赖关系集合和候选路径构建模块返回的可复用 VNF 顺序集合.对每一对依赖关系进行判断,若无冲突则进行 VNF 部署,若出现可复用 VNF 顺序与依赖关系的顺序不一致的情况,即发生冲突.发生冲突后,根据依赖关系冲突避免策略进行可复用 VNF 的取舍.

4.3.1 依赖关系冲突检测机制

获取 SR 请求中的 VNF 依赖关系集合,将提取出来的 VNF 依赖关系放到 O_{depend} 中,然后获取候选路径中可复用的 VNF 顺序的集合 O_{road} 以及需要部署的 VNF 集合 $VNFsSet_{deploy}$,其初始值计算方法如式(14)所示:

$$VNFsSet_{deploy}=Mf-(O_{road}+O_{depend}) \quad (14)$$

然后,将 O_{depend} 集合中每一对依赖关系拆分为一到多个二元组并与 O_{road} 中的顺序进行定位对比,根据冲突发生情况将发生冲突的依赖关系放到相应的冲突集合中.

获取流特征提取模块给出的服务请求中所需的 VNF 集合 Mf、服务请求中的 VNF 依赖关系 O_{depend} 以及候选路径中已存在的 VNF 集合 VNF_{c_road} 作为算法输入.通过对比候选路径中已经存在的物理序列 VNF 集合 VNF_{c_road} 与服务请求中所需的 VNF 集合 Mf,得出当前候选路径中可复用的 VNF 集合 O_{road} 以及需要部署的 VNF 集合 $VNFsSet_{deploy}$.同时,将服务请求中的依赖关系 O_{depend} 进行二元组划分,存储在依赖关系二元组 two_tuples 中.将可复用 VNF 集合 O_{road} 与依赖关系二元组 two_tuples 进行冲突对比,根据上述划分规则进行等价类划分,最终输出冲突集合 $Conflict_{set}$、可复用集合 O_{road}、需要部署集合 $VNFsSet_{deploy}$.根据上述规则,设计依赖冲突检测算法的时间复杂度为 $O(n)$,算法描述如算法 2 所示.

算法 2. 依赖冲突检测(RCD).

输入:请求中的依赖关系 O_{depend}、请求中的 VNF 集合 Mf、路径中已存在的 VNF 集合 VNF_{c_road}

输出:冲突集合 $Conflict_{set}$、可复用集合 O_{road}、需要部署的 VNF 集合 $VNFsSet_{deploy}$

```
BEGIN
1. 根据 VNF_c_road 和 Mf 集合,计算可复用 VNF 集合
   O_road
2. 根据 VNF_c_road、Mf 与 O_depend 集合,计算需要部署的
   VNF 集合 VNFsSet_deploy
3. 将请求中的依赖关系序列进行二元组划分,存入二
   元组集合 two_tuples 中
4. WHILE (i from 0 to two_tuples.size()) DO
5.   IF 二元组中两个 VNF 均在重复集合中 THEN
6.     获取第一个 VNF 在路径中位置 index1
7.     获取第二个 VNF 在路径中位置 index2
8.     IF index1>index2 THEN
9.       将该二元组存入冲突集合
10.    END IF
11.  ELSE
12.    将该二元组放入冲突集合,便于确定部署顺序
```

13.　END IF
14. END WHILE
END

4.3.2　依赖关系冲突避免

对冲突集合中的二元组进行冲突分析，根据其特点可将其进行等价类划分，可能产生的冲突种类一共有四种：

(1) 一对多：$VNF_1 \rightarrow VNF_2$，$VNF_1 \rightarrow VNF_3$；

(2) 多对一：$VNF_2 \rightarrow VNF_1$，$VNF_3 \rightarrow VNF_1$；

(3) 一对一：$VNF_1 \rightarrow VNF_2$，$VNF_3 \rightarrow VNF_4$；

(4) 传递依赖：$VNF_2 \rightarrow VNF_1$，$VNF_1 \rightarrow VNF_3$.

依赖关系冲突检测模块给出的冲突集合是这四种冲突类型的混合，本文称为交叉依赖集合. 在进行冲突处理前，需先将交叉依赖集合根据以上四种冲突类型进行划分，针对不同的冲突类型给出冲突解决策略. 下面分别对四种依赖关系进行分析：

(1) 对于一对多关系，提取共同的产生依赖的 VNF，将其标记到序列末端；

(2) 对于多对一关系，提取共同的被依赖的 VNF，将其标记到序列前端；

(3) 对于一对一或传递依赖关系，根据依赖关系定义排序规则，给出最终部署顺序.

首先，将冲突集合中的二元组进行等价类划分；然后，根据每一类的标记规则进行标记与调整，将不同类型冲突的解决方案进行归并；最终，给出所有的不可复用的依赖关系需要部署的相对顺序，将其存入冲突解决集合 O_{deploy}. 将 O_{deploy} 与可复用集合 O_{road} 进行对比，给出最终可复用集合 $VNFsSet_{reuse}$、需要部署的 VNF 集合 $VNFsSet_{deploy}$、含依赖关系需要部署的 VNF 集合 $deploySet$. 根据上述思路，设计依赖冲突避免算法的时间复杂度为 $O(n)$，算法描述如算法 3.

算法 3.　依赖冲突避免(RCA).

输入：冲突集合 $Conflict_{set}$、可复用集合 O_{road}、需要部署的 VNF 集合 $VNFsSet_{deploy}$

输出：部署的 VNF 集合 $VNFsSet_{deploy}$、依赖关系中需要部署的 VNF 集合 $deployset$、可复用 VNF 集合 $VNFsSet_{reuse}$

BEGIN
1. 从冲突集合 $Conflict_{set}$ 中导出自动扩展序列规则
2. 定义等价类划分规则
3. 生成自动扩展排序规则
4. WHILE (i from 0 to $Conflict_{set}.size()$) DO
5.　　划分并生成等价类集合
6. END WHILE
7. WHILE (i from 0 to 4) DO
8.　　分别生成等价类序列 $flag==classn(n \in 1 \sim 4)$
9. END WHILE
10. IF $flag==class1$, THEN
11.　　提取共同的产生依赖的 VNF，将其标记到其他所有 VNF 的末端
12. ELSE IF $flag==class2$, THEN
13.　　提取共同的被依赖的 VNF，将其标记到其他所有 VNF 的前端
14. ELSE
15.　　输入排序规则，导出最终序列
16. END IF
17. 归并四种等价类序列，给出冲突集合 VNF 部署顺序 O_{deploy}
18. 更新候选路径中复用 VNF 集合 $VNFsSet_{reuse}$ 中
19. 更新依赖关系中需要部署 VNF 集合 $deployset$
20. 更新需要部署 VNF 集合 $VNFsSet_{deploy}$
END

4.4　VNF 部署

由于在部署过程中 VNF 会改变数据量的大小，LFGL 原则被广泛使用. LFGL 原则可以在流量路径上实现最优链路负载[13]. 当服务流的最终路径已经构建完成，且仅剩下部署 VNFs 的任务，若为满足服务需求需要在最终路径部署所需 VNFs 时，可采用 LFGL 原则对 VNFs 进行部署[9].

LFGL 原则首先根据所有 VNF 的流入流出比对其进行升序排序. 然后，将带有非正因素的中间件放置在路径的开头，或者说是将 VNF 以流量变化因子递增的顺序从路径的源节点逐一部署. 当一个存储服务器用完其空间容量时，继续在路径上查找下一个服务器节点继续进行部署. 减少流量的 VNF 部署完成后，该规则将切换到对流量有扩展作用的中间件，并将它们从路径目的节点按其流量变化因素的降序排列. 如果路径中的服务器节点可将所有的 VNF 部署完成，则部署成功，否则，部署失败.

本文基于 LFGL 原则对不同集合中的 VNF 进行部署. 在依赖冲突解决之后，可给出相应的服务功能链及严格的 VNF 部署顺序，此时，根据 VNF 部署序列要求完成相应 VNF 的部署. 在进行部署时，首先，根据 LFGL 原则将 $VNFsSet_{deploy}$ 集合中的 VNF 部署完成，以保证最大化链路剩余带宽；然后根据依赖关系顺序要求完成 $deploySet$ 中 VNF 的定向部署. 需要说明的是：根据候选路径选取规则可知，该路径的剩余资源可满足服务请求中所有 VNF 的资源需求. VNF 部署算法的时间复杂度为 $O(n^2)$，算法描述如算法 4 所示.

算法 4. VNF 部署(VD).

输入:候选路径 $CandidatePath$,需要部署 VNF 集合 $VNFsSet_{deploy}$,依赖关系中需要部署的 VNF 集合 $deployset$

输出:VNF 嵌入图

```
BEGIN
1. 将 VNFsSet_deploy 集合中 VNF 根据流入流出比进行排序;
2.    WHILE (i from 0 to VNFsSet_deploy.size()),DO
3.       WHILE 流入流出比小于 1,DO
4.          WHILE (j from 0 to CandidatePath.size()),DO
5.             IF 当前节点剩余资源满足请求所需资源,THEN
6.                在该服务器部署 VNF 并进行标记;
7.                更新部署 VNF 的服务器的剩余资源;
8.             ELSE
9.                j++;
10.            END IF
11.         END WHILE
12.      END WHILE
13.      WHILE 流入流出比大于等于 1,DO
14.         WHILE (k from CandidatePath.size() to j)
15.         IF 当前节点剩余资源满足请求所需资源,THEN
16.            在该服务器部署 VNF 并进行标记;
17.            更新部署 VNF 的服务器的剩余资源;
18.         ELSE
19.            k--;
20.         END IF
21.      END WHILE
22.   END WHILE
23. END WHILE
24. IF i<VNFsSet_deploy.size(), THEN
25.      舍弃该流;
26.      EXIT //退出程序;
27. ELSE
28.      获取当前链路 VNF 物理序列;
29.      WHILE (i from 0 to deployset.size()),DO
30.         获取依赖关系;
31.         定向部署 deployset 中的 VNF;
32.      END WHILE
33.      IF i<deployset.size(), THEN
34.         舍弃该流;
35.         EXIT //退出程序
36.      END IF
37. END IF
END
```

5 仿真实现与性能评价

5.1 仿真环境

该仿真实验的实验环境参数为 CPU:Intel(R) Core(TM) i7-1065G7 CPU@1.30 GHz 1.50GHz,GPU:NVIDIA GeForce GTX 1050Ti,16 GB 内存,1 TB 硬盘,Windows 10(64)位操作系统,编程环境为:IntelliJ IDEA.

5.2 仿真实现

5.2.1 实验拓扑

为了验证本文设计的基于 VNF 的网络服务冲突检测与避免机制(VNF-based Network Service Conflict Detection and Avoidance Mechanism,CDA)的性能,本文在不同规模的拓扑下进行实验,小规模网络选择 BT-Europe 拓扑,大规模网络选择 Forthnet 拓扑. 图 4 为 BT-Europe 主干网的拓扑图,该拓扑共有 21 个节点,29 条边. 图 5 为 Forthnet 主干网的拓扑图,该拓扑共有 60 个节点,57 条边.

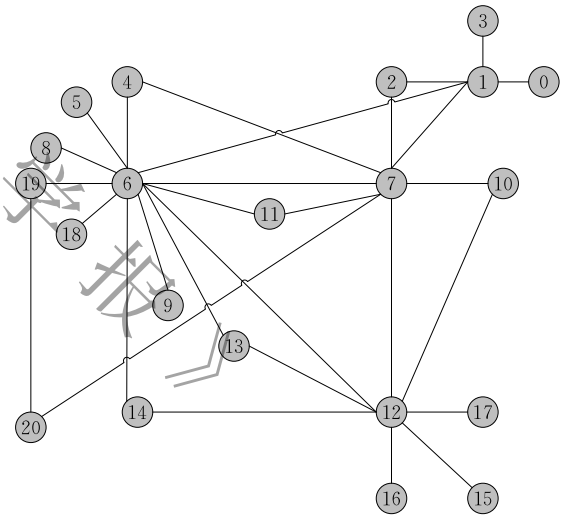

图 4 BT-Europe 拓扑图

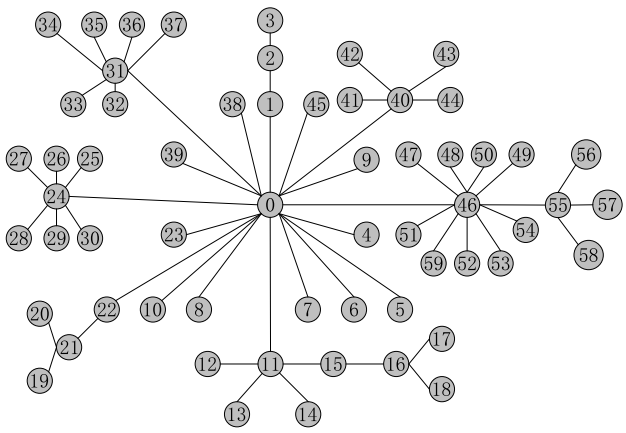

图 5 Forthnet 拓扑图

5.2.2　参数设计

本文基于 Java 语言搭建基础的 NFV 网络服务平台，用于提供网络服务. 并且在该平台上对本文所提出的机制进行了开发，NFV 基础网络平台的相关参数[14]设计如表 1 所示.

表 1　平台参数设置

参数名称	分布	取值
VNF 种类数量	无	50 种
单条请求包含 VNF 个数	均匀分布	2～10 个
单个 VNF 的 CPU	均匀分布	1～20 GHz
单个 VNF 所需存储	均匀分布	1～20 G
VNF 流入流出比	均匀分布	0.5～2
请求生成时间	泊松分布	4req/100unit
单个请求 CPU	均匀分布	20～40 GHz
单个请求带宽	均匀分布	1～30 Mbps
请求初始数据量	均匀分布	25～30 M
每个依赖中 VNF 数量	均匀分布	2～4 个/依赖
单条请求分解为包的数量	均匀分布	2～5 个/流
请求生存周期	指数分布	均值 1000 unit
节点 CPU	均匀分布	100～150 GHz
节点存储容量	均匀分布	100～150 G
最大允许时延	无	1000 unit

5.3　性能评价

5.3.1　对比机制

基于 VNF 的网络服务冲突检测与避免机制，综合考虑了 VNF 复用性与依赖性的冲突. 选取 Fang 等人在文献[15]中提出的最长公共子序列部署(the Longest Common Subsequence Based Algorithm，LBA)算法以及 Xu 等人在文献[16]中提出的(Coordinated Chaining and Mapping Algorithm，CCMA)算法作为对比实验.

5.3.2　评价指标

本文将通过平均复用率、SFC 平均时延及时延标准差、SFC 部署成功率、冲突率和冲突成功避免率五个指标，对基于 VNF 的网络服务冲突检测与避免机制及其对比实验的性能进行评价. 现对指标进行介绍：

(1) 平均复用率

$$R=\frac{\sum_{i=0}^{req_{num}} S*\frac{reuse_{num}}{VNF_{num}}}{deploy_{success}} \tag{14}$$

其中，S 为状态标记，$S=1$ 代表该请求被成功部署，$S=0$ 代表该请求未成功部署. $reuse_{num}$ 代表当前请求中 VNF 的复用数量. VNF_{num} 代表当前请求中 VNF 的需求总数量. $deploy_{success}$ 代表所有请求中被成功部署的请求的数量.

本文所提出机制的目标为复用率最大，在进行了平均复用率对比后，可利用累积分布函数(Cumulative Distribution Function，CDF)来进一步验证不同机制下复用率的概率分布. 而各个机制中不同负载下的复用率数值表现为离散型，所以对于不同机制下 CDF 的表现，代表了小于等于复用率的值出现的概率之和.

$$F_X(x)=P(X\leq x) \tag{15}$$

其中，$F_x(x)$ 代表概率的和，P 代表概率值，x 为复用率的取值.

(2) SFC 平均时延

$$\varnothing=\frac{\sum_{i=1}^{|Mf|}\frac{CPUofreq}{CPUofVNF}+\frac{init*\prod_{i=1}^{path,size} radio_i}{BandWidth_i}}{deploy_{success}} \tag{16}$$

其中，$CPUofreq$ 代表请求所需处理能力. $CPUofVNF$ 代表 VNF 所能提供的处理能力. $init$ 代表当前请求的初始数据量. $radio_i$ 代表当前链路已部署的 $VNFi$ 的流入流出比. $BandWidth_i$ 代表当前链路带宽. $deploy_{success}$ 代表所有请求中，被成功部署的请求的数量.

(3) SFC 部署成功率

$$\alpha=\frac{deploy_{success}}{sum} \tag{17}$$

其中，$deploy_{success}$ 代表被成功部署的请求数量. sum 代表网络发出的请求的总数量.

(4) 冲突率

$$\beta=\frac{Conflict\_num}{sum} \tag{18}$$

其中，$Conflict\_num$ 代表当前时刻请求中发生冲突的流的个数，sum 代表当前时刻网络内所有的流的数量.

(5) 冲突成功避免率

$$x=\frac{deploy_{success}-deploy_{unconflict}}{Conflict\_num} \tag{19}$$

其中，$deploy_{success}$ 代表被成功部署的流的数量. $deploy_{unconflict}$ 代表不产生冲突时，网络中可被成功部署的流的数量，$Conflict\_num$ 代表当前网络中产生冲突的流的个数.

5.3.3　性能评价

(1) 平均复用率

本文所提出机制的目标函数为复用率最大. 复用率越高，网络中已部署的 VNF 的利用率越大，可以减少重新部署 VNF 的开销与维护冗余 VNF 运行的开销. 图 6 与图 7 分别反映了该机制在 BTEurope 和 Forthnet 两种不同规模拓扑下的运行效果.

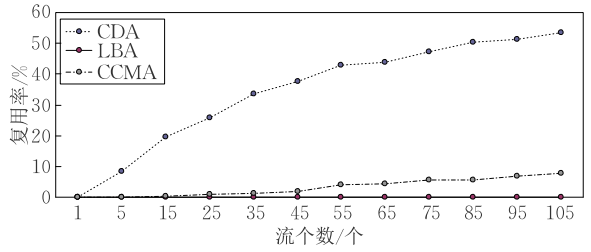

图 6 BTEurope 拓扑中平均复用率

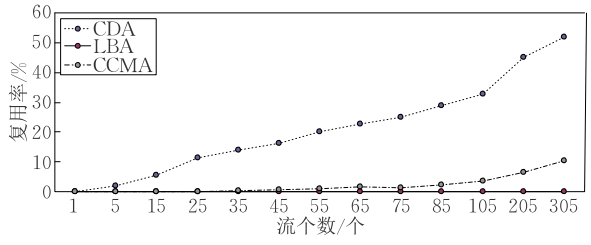

图 7 Forthnet 拓扑中平均复用率

从图 6 与图 7 中可以看出,三个机制中 LBA 表现最差,因为该机制在整个的构建和映射过程中不考虑 VNF 的可复用性,所以其复用率为 0. 本文提出的 CDA 算法在复用率方面的总体性能表现相对于 CCMA 较优. 因为本文机制在构建 SFC 之前优先考虑复用性与依赖性之间的冲突,在保证复用率最大的前提下解决二者之间的冲突并完成 SFC 构建,所以其复用率较高. 而 CCMA 是在部署时,若遇到可复用 VNF 则复用,未将其作为先导条件进行考虑,所以其复用率较低且增速较慢.

(2)平均时延

时延是评价网络性能的重要指标,图 8 与图 9 分别反映了各机制在 BTEurope 和 Forthnet 两种拓扑下,以及不同负载情况下的时延表现.

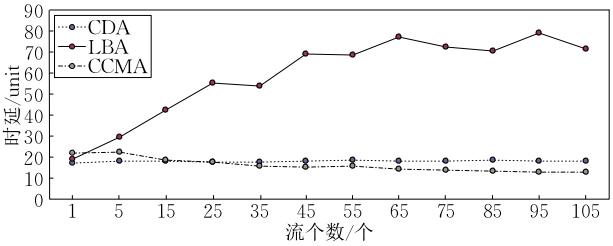

图 8 BTEurope 拓扑下时延对比

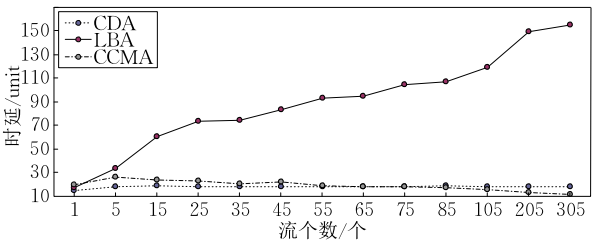

图 9 Forthnet 拓扑下时延对比

从实验结果可以看出,在数据请求较少时,所提出的 CDA 机制最优,其次为 LBA,而 CCMA 的表现性能最差. 随着请求的增多,CDA 时延没有大幅提升,LBA 时延增大且在小型拓扑中不断波动,CCMA 表现性能越来越好. 而数据量较大时,CCMA 机制优于 CDA,LBA 表现最差. CDA 和 CCMA 虽然机制不同,但均对 VNF 的部署顺序及链路带宽进行了考虑,所以其时延较小. 在数据流较少时 CDA 表现较好,数据流变大后 CDA 表现较 CCMA 弱,是因为 CDA 综合考虑了 VNF 的复用性与依赖关系的冲突,在前期冲突较少,SFC 构建顺序拟合于 LFGL 原则,保证了最大化链路剩余带宽,并且在构建候选路径时考虑了带宽的影响,选择的路径较优,所以总时延较少. 而随着数据量增大,复用性与依赖关系的冲突展现出来,在保证复用率的同时一定程度地削弱了 LFGL 对最大化链路带宽的优化,导致了时延较 CCMA 大. 但总体来讲,在数据量不断增大时 CDA 与 CCMA 的差距并未出现显著增加,且明显优于 LBA 算法.

(3)部署成功率

从图 10 和图 11 可以看出,在数据请求较少时,三者表现性能相同,均能成功部署全部的数据请求. 而随着数据量的增大,一定负载内 CDA 与 CCMA 成功部署率相当,随着负载继续增大,LBA 成功率最先开始下降,且下降速度较快. 当数据量持续增大时,CDA 与 CCMA 也出现下降且 CCMA 下降速率更快,CDA 的成功部署率始终优于 CCMA. LBA 在提供服务过程中对 VNF 的复用性及带宽考虑不足,所以在出现大量请求时,网络中可复用 VNF 的资源未被合理利用,从而导致大量资源浪费,且 VNF 对数据量的影响未被考虑,导致传输时延增大,部分流被抛弃. 而 CDA 和 CCMA 考虑链路 VNF 的复用,并且基于 LFGL 原则对链路剩余带宽进行考量,所以二者性能明显优于 LBA. 而 CDA 在构建路径时将复用作为约束条件,即所选路径复用率较大,将复用率作为先导条件. 而 CCMA 只是在构建路径

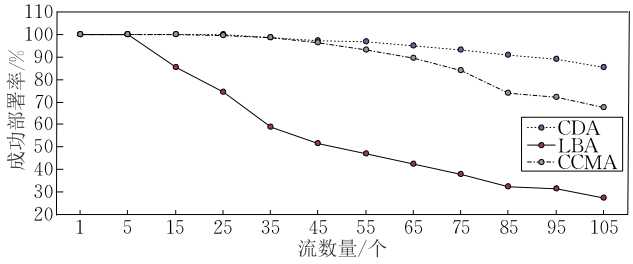

图 10 BTEurope 拓扑下成功部署率

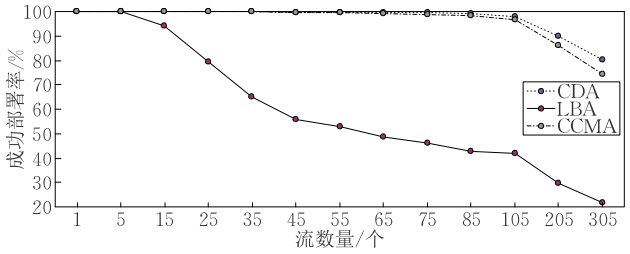

图 11　Forthnet 拓扑下成功部署率

后检查是否有可复用 VNF，所以其复用率较小. 当流增多时，CDA 的复用率更高，更节约网络资源，剩余网络资源可以为其他请求提供服务，所以其部署率更高.

(4) 冲突率及冲突成功避免率

本文通过测试冲突率检测网络冲突存在的概率，并通过测试冲突成功避免率检测本机制. 图 12 与图 13 反映了 CDA 机制在 BTEurope 与 Forthnet 两种拓扑下，随负载的增加冲突的发生率以及冲突成功避免率的变化情况. 冲突率计算公式如式(19)所示，冲突避免率计算公式如式(20)所示. 这里需要说明的是，图 12 和图 13 均为多次实验的平均值，即此时的冲突率计算公式如式(21)所示，冲突避免率公式如式(22)所示. 其中，n 为独立实验的次数.

$$\bar{\beta}=\frac{\sum_{i=0}^{n}\frac{Conflict\_num}{sum}}{n} \tag{20}$$

$$\bar{x}=\frac{\sum_{i=0}^{n}\frac{deploy_{success}-deploy_{unconflict}}{Conflict\_num}}{n} \tag{21}$$

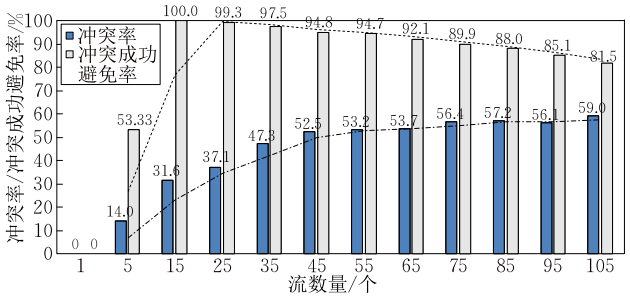

图 12　BTEurope 拓扑下冲突率及冲突成功避免率

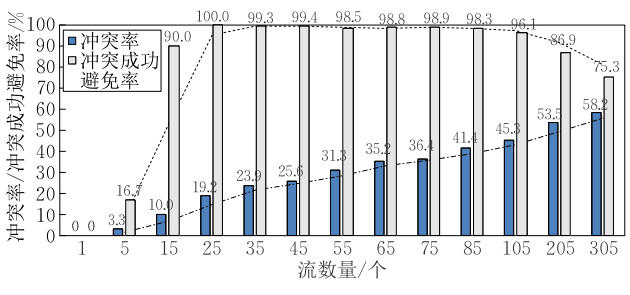

图 13　Forthnet 拓扑下冲突率及冲突成功避免率

从图 12 和图 13 中可以看出，在网络内只有 1 个数据包时，网内未发生冲突，所以冲突成功避免率为 0. 而在负载较少时，网络冲突发生率较低，冲突均能被成功避免，单个样本的冲突成功避免率为 100%，但实验中部分未发生冲突的请求的冲突成功避免率为 0，所以导致样本总体的平均冲突解决率较低. 随着负载持续增大，冲突率不断上升，每一次请求发生都会产生冲突，在一定范围内其冲突避免率稳定在 100%左右. 当负载不断增大，冲突率也随之增大，但网络资源有限，所以冲突成功避免率也随着数据包被抛弃而减少. 综上所述，该机制的冲突检测率以及冲突避免率表现较优.

6　结束语

本文对基于 NFV 的现有研究工作进行了调研，发现其对资源分配问题的研究主要集中于转发图嵌入方面，而对转发链构建和虚拟网络功能调度方面的研究相对较少. 并且，在网络服务请求到达后，在 SFC 的构建以及 VNFs 的部署中存在一些特征问题，已有学者针对这些特征问题展开研究，但只在 VNF 链路构建与部署时考虑单个影响因素，未对其进行综合考虑. 本文根据调研产生的特征问题，进行网络服务冲突种类构建. 并将网络服务流程进行拆解，通过不同模块串联，在不同模块进行相应的冲突检测与避免，以提高网络性能. 实验结果表明，本文设计的基于 VNF 的网络服务冲突检测与避免机制在复用率、时延、部署成功率方面所表现出的性能均优于对比算法.

参　考　文　献

[1] Sun P, Lan J, Li J, et al. Combining deep reinforcement learning with graph neural networks for optimal VNF placement. IEEE Communications Letters, 2021, 25(1): 176-180

[2] Gupta A, Jaumard B, Tornatore M, Mukherjee B, et al. A scalable approach for service chain mapping with multiple SC instances in a wide-area network. IEEE Journal on Selected Areas in Communications, 2018, 36(3): 529-541

[3] Chen Y, Wu J, Ji B. Deploying virtual network functions with non-uniform models in tree-structured networks. IEEE Transactions on Network and Service Management, 2020, 17(4): 2260-2274

[4] Wang Z, Zhang J, Huang T, et al. Service function chain composition, placement and assignment in data centers. IEEE

Transactions on Network and Service Management, 2019, 16(4): 1638-1650

[5] Ljubi I, Mouaci A, Perrot N, et al. Benders decomposition for a node-capacitated virtual network function placement and routing problem. Computers & Operations Research, 2021, 130: 105227

[6] Hawilo H, Jammal M, Shami A. Network function virtualization-aware orchestrator for service function chaining placement in the cloud. IEEE Journal on Selected Areas in Communications, 2019, 37(3): 643-655

[7] Beck M T, Botero J F. Coordinated allocation of service function chains. Network Function Virtualization & Software Defined Networks. Berlin, Germany, 2017: 78-88

[8] Kuo T, Liou B, Lin K C, Tsai M. Deploying chains of virtual network functions: On the relation between link and server usage. IEEE/ACM Transactions on Networking, 2018, 26(4): 1562-1576

[9] Xu Z, Jin C, Yang F, et al. Coordinated resource allocation with VNFs precedence constraints in Inter-Datacenter Networks over elastic optical infrastructure//Proceedings of the 2018 IEEE World Symposium on Communication Engineering (WSCE). Singapore, 2018: 1-6

[10] Ning Zili, Wang Ning, Tafazolli R. Deep reinforcement learning for NFV-based service function chaining in multi-service networks//Proceedings of the 2020 IEEE 21st International Conference on High Performance Switching and Routing (HPSR). Denver, USA, 2020: 1-6

[11] Nejad M A T, Parsaeefard S, Maddah-Ali M A, et al. vSPACE: VNF simultaneous placement, admission control and embedding. IEEE Journal on Selected Areas in Communications, 2018, 18(4): 542-557

[12] Lukovszki T, Rost M, Schmid S. Approximate and incremental network function placement. Journal of Parallel and Distributed Computing, 2018, 120: 159-169

[13] Ma W, Medina C, Pan D. Traffic-aware placement of NFV middleboxes//Proceedings of the Global Communications Conference. San Diego, USA, 2015: 1-6

[14] Sahhaf S, Tavernier W, Rost M, et al. Network service chaining with optimized network function embedding supporting service decompositions. Computer Networks, 2015, 93(DEC. 24): 492-505

[15] Fang W, Zeng M, Liu X, et al. Joint spectrum and IT resource allocation for efficient VNF service chaining in inter datacenter elastic optical networks. IEEE Communications Letters, 2016, 20(8): 1539-1542

[16] Yang S, Li F, Trajanovski S, et al. Recent advances of resource allocation in network function virtualization. IEEE Transactions on Parallel and Distributed Systems, 2021, 32(2): 295-314

LI Zheng-Yu, M. S. candidate. His research interest mainly include named data networking, software defined networking, traffic engineering, artificial intelligence.

HUANG Min, Ph. D., professor, Ph. D. supervisor. Her research interest mainly include intelligent optimization algorithm, scheduling theory and methods etc.

YI Bo, associate professor. His research interest mainly include network function virtualization and service chain etc.

WANG Xing-Wei, Ph. D., professor, Ph. D. supervisor. His research interest mainly include next generation internet, cloud computing, network security and information security etc.

Background

Under the traditional network architecture, service providers need to deploy a large number of proprietary devices (such as middleware) in the network to meet user needs. The deployment of middleware has led to high operating expenditure and cost expenditure, and has also made the traditional network bloated and rigid. Network Functions Virtualization (NFV) replaces existing network functions in the form of software, which is called Virtual Network Function (VNF). NFV architecture realizes the decoupling of software and hardware, which makes the network function update more flexible while reducing expenditure. However, while NFV improves network flexibility, it also brings new challenges, such as how to build a service function chain (SFC) and allocate resources to provide users with satisfactory network services.

With the rapid increase of network service functions, people put forward higher requirements for network services. The emergence of NFV architecture has solved this problem to some extent, effectively reducing the cost of service providers, shortening the deployment cycle, and also increasing the flexibility of the network. Although various fields have been trying to make NFV a reality, there are still some

unresolved problems, such as VNF lifecycle management and orchestration. Improving resource utilization and reducing VNFs deployment cost and overhead cost is also a concern. Because VNFs have life cycles, when there are a large number of deployed VNFs in the network, for newly arrived requests, you can make a comprehensive judgment based on the remaining resources, remaining life cycles, and network functions included in the request of the deployed VNFs, and build a path with high reuse rate for them to improve the reuse rate of deployed VNFs in the current network, thus effectively improving the resource utilization. At the same time, when the network processing capacity is unchanged, it can also effectively improve the network throughput, and reduce the deployment cost of repeatedly deploying VNFs and the overhead cost caused by running the same VNFs. In the process of building a path with high reuse rate, because the service request contains the dependencies of VNFs, and the deployed VNFs have physical sequence restrictions, VNFs dependency conflicts are likely to occur.

In order to effectively improve the reuse rate of VNFs, reduce the deployment cost and overhead cost of VNFs, efficiently use network resources and improve network throughput, this paper conducts conflict detection and avoidance for VNFs based network services. When a network service request arrives, a candidate path with the highest reuse rate and the lowest cost is constructed for it through a series of restrictions. The deployed VNFs on the path are adjusted according to the VNF dependency in the request, so as to ensure the maximum reuse rate while meeting the request dependency. Due to the different order of VNF, the data volume will change. After the VNF deployment order is adjusted, it is necessary to judge whether the path meets the bandwidth limit. When the candidate path cannot meet the current service request, it is necessary to dynamically adjust the path.

This work is supported by the National Key R&D Program of China under Grant No. 2019YFB1802800, the National Natural Science Foundation of China under Grant. Nos. 62002055, 62032013, 61872073.