DOI:10.13718/j.cnki.xsxb.2019.06.017

一种虚拟化网络功能启发式动态编排算法①

母泽平

重庆电子工程职业学院 人工智能与大数据学院,重庆 401331

摘要:随着信息化的发展,网络业务的种类越来越多,业务的功能越来越强大,网络的基础设施为业务提供动态服务的能力已跟不上业务发展的速度,研究动态部署虚拟化网络功能具有重大意义.在不违反服务水平协议的情况下,研究了虚拟网络功能编排问题,并提出了虚拟网络功能编排的整数线性规划数学模型,接着基于动态编程的启发式算法对模型求解,最后对现实世界网络拓扑进行跟踪模拟.仿真结果表明,所提出的启发式算法可以降低网络运营成本,相关性能优于传统的硬件中间件方法.

关 键 词:网络功能虚拟化;网络功能编排;启发式算法

中图分类号:TP393 **文献标志码**:A **文章编号**:1000-5471(2019)06-0092-11

当今网络系统为各种业务提供各种不同的服务,需要在垂直式架构上部署各种专有中间件或网络设备(包括防火墙、代理、WAN 优化器、入侵检测系统和入侵防御系统)来实现各种性能和安全目标[1-2].文献[2-3]研究表明,中间件的数量与数据中心路由器的数量相当.即使中间件已经成为现代网络的一个组成部分,但它们具有高资本支出和运营支出,它们通常是供应商特定的、垂直整合的、昂贵的,并且需要经过专门培训的人员进行部署和维护.此外,向现有的中间件添加新功能是困难的,这使得网络运营商部署新服务变得非常困难和麻烦,网络运营商在引入新网络服务时,被迫升级或购买新硬件.

同时,多网络以特定顺序处理负载流,在不同阶段由不同的中间件处理才能完成,例如可能需要先通过防火墙,然后进入入侵检测系统,最后通过代理服务器[4],这种现象对于中间件是非常常见的,通常被称为作为服务功能链[5].IETF 网络和服务链工作组提出了 IETF 运营商网络中的中间件链接的案例,如移动网络[6]和数据中心网络[7]案例,对这些网络中间件处理进行排序的任务通常被称为网络功能编排.目前,中间件放置在网络中的固定位置,通过手工制作路由表条目,实现中间件序列路由负载流,这是一个麻烦且容易出错的过程,而且,长期固定中间件的位置对于所有可能的负载流模式来说不一定是最佳的.

网络功能虚拟化对通用的设备在网络服务器上虚拟成各种不同的虚拟化网络功能,从而解决动态局限性[8-9].它将数据包处理从硬件中间件迁移到服务器的软件上运行,这种方法不妨碍性能,文献[10-11]表明软件中间件已经近似实现硬件性能.网络功能虚拟化可以优化网络、降低成本.以前,相关专用硬件设备放置在固定位置,现在则是在网络中的任何服务器上部署虚拟化的网络功能进行替代,从而智能地确定虚拟化的网络功能位置,以确保有效的负载流路由.网络功能虚拟化提供了优化虚拟化的网络功能位置和流量路由路径的机会,这样显著地减少了网络营运成本.

在单个服务器或服务器集群上动态部署虚拟化的网络功能链,能显著降低网络的部署成本.然而,在配置虚拟化的网络功能之前需要考虑几个问题:

1)部署新虚拟化的网络功能的成本;

① 收稿日期:2018-12-10
基金项目:第五批重庆市高校优秀人才项目(2017.29);重庆电子工程职业学院 2017 年度校级科研平台项目(XJPT201707).
作者简介:母泽平(1965-),男,讲师,本科,主要从事计算机科学与应用研究.

2）运行虚拟化的网络功能的能源成本；

3）虚拟化的网络功能转发流量的成本；

4）底层物理资源池的碎片化.

从以上问题可以发现，如果只考虑足够的虚拟化的网络功能来满足流量处理要求，这样将会产生最低的部署和能源成本，然而，可能会增加流量转发成本，并可能最终导致服务级目标违规. 另一方面，通过部署虚拟化网络功能始终通过最短的路径转发流量位置，这种方法可能会避免服务级目标违规处罚，但会导致部署和能源成本的增加. 在成本最小化过程中，最佳虚拟化的网络功能业务流程策略必须解决这些问题. 此外，它必须避免服务级别目标违规，并满足物理服务器和物理链路的容量限制.

1　相关研究

1.1　网络功能的管理及编排

早期的虚拟网络功能管理，主要集中在云服务的包管理方法方面[2]. 这种包的管理是通过研究不同网络运营商的经验来减轻当今企业网络中出现的管理复杂性. Stratos[13] 和 OpenNF 等[14] 提出了对 NFV 的更进式的管理方法. Stratos 通过处理负载工程，提出了水平缩放等方式将虚拟化的网络功能存放在远程云平台中. OpenNF 通过扩展集中式 SDN 模式为虚拟化的网络功能和网络转发平面提出了融合控制平面[12]. 最近关于虚拟化的网络功能管理的一些工作侧重于流量工程问题，例如通过一些现有的虚拟化的网络功能序列来引导流量[4]. 当一些虚拟化的网络功能修改分组报头时，从而改变流量特征，这个问题变得更具挑战性. Qazi[4] 和 Fayazbakhsh 提出了基于标记的机制，以在其生命周期中识别流量，并且还跟踪虚拟化的网络功能的访问序列. 这些工作专注于确定虚拟化的网络功能链的布局和流量路由路径，而基于标记的方法可用于在虚拟网络中部署虚拟化的网络功能链.

1.2　虚拟功能及虚拟功能链的部署

Mehraghdam 等[13] 提出了一种用于指定虚拟化的网络功能链的方法，同时提出了虚拟化的网络功能链放置的数学模型. 然而，求解方法是非线性的，不允许多个租户之间的虚拟化的网络功能共享. 文献[14]基于 LP 松弛，提出了一种用于查找数据间中心虚拟化的网络功能链放置的方法，然而，由于 LP 松弛，文献[14]上限超过了低层最多 16 个物理资源的能力. 文献[15]提出了自动化虚拟化的网络功能布局的编排架构，但作者没有提供任何具体的编排算法.

2　系统模型

2.1　物理网络

将物理网络表示为无向图 $\bar{G}=(\bar{S},\bar{L})$，$\bar{S}$ 表示交换机的集合，$\bar{L}$ 表示链路的集合. 假设虚拟网络功能可以部署在位于网络内的服务器上，这些网络位置传统上称为“点”$\bar{n}$. 集合 $\bar{N}$ 用二进制变量 $\bar{h}_{\bar{n}\bar{s}}\in\{0,1\}$ 代表这些服务器 $\bar{n}\in\bar{N}$ 是否附加到交换机 $\bar{s}\in\bar{S}$，如(1)式所示：

$$\bar{h}_{\bar{n}\bar{s}}=\begin{cases}1 \text{ if server } \bar{n}\in\bar{N} \text{ is attached to switch } \bar{s}\in\bar{S}\\ 0 \text{ otherwise}\end{cases} \tag{1}$$

令 R 表示由每个服务器提供资源集合(CPU，内存，磁盘等)，$c_{\bar{n}}^{r}\in\mathbb{R}^{+}$ 表示服务器 $\bar{n}$ 的资源容量，$\beta_{\bar{u}\bar{v}}\in\mathbb{R}^{+}$，$\delta_{\bar{u}\bar{v}}\in\mathbb{R}^{+}$ 表示带宽容量和物理链路 $(\bar{u},\bar{v})\in\bar{L}$ 的传播延迟，定义 $\eta(\bar{u})$ 作为交换机 $\bar{u}$ 的集合，则存在(2)式的约束关系.

$$\eta(\bar{u})=\{\bar{v}\mid(\bar{u},\bar{v})\in\bar{L}\text{ or }(\bar{v},\bar{u})\in\bar{L}\},\ \bar{u},\bar{v}\in\bar{S} \tag{2}$$

2.2　虚拟化网络功能

不同类型的虚拟网络功能集在现实网络中进行配置，虚拟网络功能的类型用集合 p 表示. 用 $D_{\mathfrak{h}}^{+}$，$\kappa_{\mathfrak{h}}^{r}\in\mathbb{R}^{+}(\forall r\in R)$，$c_p$(Mbps)，$\delta_p$(ms) 表示每个虚拟功能的类型 p 的部署成本，资源需求、处理能力和处理延迟如下所述：

部署成本：在服务器上传输和部署虚拟服务功能 p 的成本.

资源需求：必须分配给类型 p 的虚拟网络功能的类别 r 的资源量.

处理能力：表示虚拟网络功能可以处理的流量(以 Mbps 为单位).

处理延迟：是通过类型 p 的虚拟网络功能时数据包遇到的平均延迟(以 ms 为单位).

上述数量的实际值取决于很多因素，在这里，假设用这些属性的近似值来简化数学模型，可能存在某些硬件要求，会阻止服务器运行特定类型的虚拟化的网络功能. 此外，网络管理员可以具有在特定服务器集上提供特定类型的虚拟化的网络功能的偏好. 因此，假设对于每个虚拟化的网络功能类型都有一组服务器可以在其上进行配置，使用下面二进制变量表示此关系：

$$d_{np}=\begin{cases}1 \text{ if } p\in P \text{ can be providioned on } \bar{n}\\ 0 \text{ otherwise}\end{cases} \tag{3}$$

2.3 负载请求

假设网络运营商正在接收用于设置不同类型业务的路径的请求. 负载请求由 6 元组 $t=\langle \bar{u}^t, \bar{v}^t, \psi^t, \beta^t, \delta^t, \omega^t\rangle$ 表示，其中 $\bar{u}^t, \bar{v}^t \in \bar{S}$ 表示入口和出口开关，$\beta^t \in \mathbb{R}^+$ 是负载需求的带宽，δ^t 是根据服务水平协议允许的最大传播延迟，ψ^t 是负载虚拟网络功能必须有序通过序列，$l\psi^t$ 是 ψ^t 的长度，ω^t 表示确定服务级目标违规的政策违规处罚.

此数学模型中，将虚拟网络功能序列 ω^t 转换为有向无环图 $G^t=(N^t, L^t)$，其中 N^t 表示一组业务节点(虚拟网络功能，入口和出口的交换机)，L^t 表示它们之间的链路，以这种方式建模负载流使得配置过程容易，以确保其通过正确的虚拟网络功能序列. 还定义 $\eta^t(n_1)$ 来表示 $n_1 \in N^t$ 的邻居：

$$\eta^t(n_1)=\{n_2 \mid (n_1, n_2)\in L^t\},\ n_1, n_2 \in N^t \tag{4}$$

不失一般性，在原有向图 Gt 中，如果 n_2 出现在 n_1 之后，考虑 $n_2 > n_1 (n_1, n_2 \in N^t)$，接着定义一个二进制变量 $g^t_{np} \in \{0, 1\}$ 表明节点 $n \in N^t$ 的类型.

$$g^t_{np}=\begin{cases}1 \text{ if node } n\in N^t \text{ is of type } p\in P\\ 0 \text{ otherwise}\end{cases} \tag{5}$$

2.4 虚拟网络功能编排问题

考虑一个运营网络正在服务一组流量的场景，它已经部署了一套虚拟化的网络功能并且还提供了业务的路由路径. 现在，网络运营商正在接收新的流量请求，并希望为它们提供所需的虚拟化的网络功能和路由路径，网络运营商可以一次选择为一个流量请求提供资源，也可以选择通过累积多个流量请求并分批提供资源来利用查找间隔. 在获得物理网络拓扑、虚拟化的网络功能规范、当前网络状态和一组新的流量请求条件下，通过测量空闲资源的百分比来计算物理资源碎片用于活动的服务器和链接，希望最大限度地减少碎片，实现提供最佳数量的虚拟化的网络功能，将其放置在最佳位置，以及为每个流量请求找到最佳路由路径，同时尊重容量约束，并确保流量通过适当的虚拟化的网络功能序列的目标.

3 问题公式化

与传统的虚拟网络嵌入问题相比，虚拟化的网络功能编排是一个 NP 难问题[16]，虚拟网络嵌入中没有节点排序要求，而在虚拟化的网络功能编排中，需要保留虚拟化的网络功能的排序. 此外，在虚拟化的网络功能编排中，需要尊重服务器和部署的虚拟化的网络功能的处理能力约束，要部署的虚拟化的网络功能数量并不是提前知道的，而是优化过程的结果. 集装箱问题可以用来解决虚拟化的网络功能编排，但是在这里将最终得到一个嵌套的集装箱问题. 在第一层中，流量需要打包成虚拟化的网络功能，下一层需要将虚拟化的网络功能打包到物理服务器中. 事实上，虚拟化的网络功能的数量和位置预先不知道，导致资源容量的二次约束.

3.1 物理网络传输

转换物理网络以生成增强的虚拟网络，从而降低虚拟网络功能编排的复杂性，在以下两个步骤中进行转换处理：

3.1.1 虚拟网络功能处理

物理网络拓扑如图 1(a)所示. 在这里，有 3 个开关($s1$, $s2$ 和 $s3$)和连接到开关 $s2$ 的服务器 $n2$，通过查找最大数量，此步骤中的所有可能的虚拟化的网络功能可以部署在任意服务器上，根据资源的能力来计算

这个数字服务器和一种虚拟化的网络功能的资源需求. 例如，如果一个服务器有 16 个内核，防火墙和 IDS 分别需要 4 核和 8 核，可以在其上部署 4 个防火墙或 2 个 IDS. 在图 2(b)中显示枚举了服务器 $n2$ 的虚拟化的网络功能.

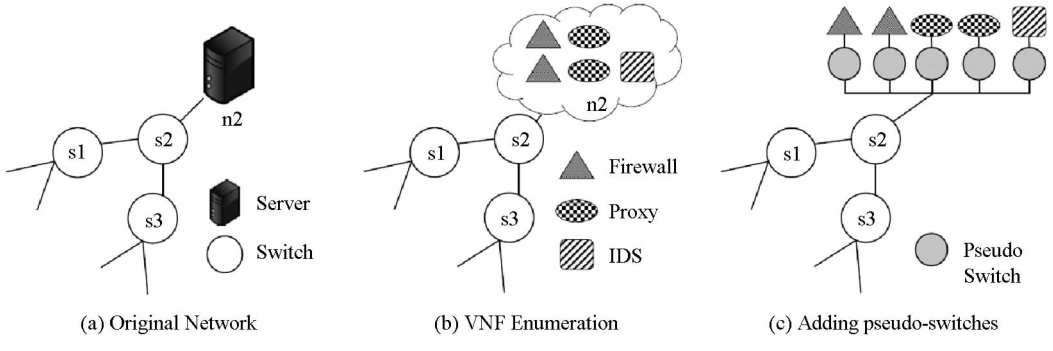

图 1　虚拟网络功能转换图

定义 M 为虚拟网络功能集，每个虚拟网络功能 $m \in M$ 隐含地附加到服务器 $\bar{n} \in \bar{N}$，还将两个附加的伪虚拟化的网络功能附加到每个服务器以表示流量的入口和出口点请求，使用函数 $\xi(m)$ 表示上面的映射.

$$\xi(m) = \bar{n} \text{ if } m \text{ is attached to server } \bar{n} \tag{6}$$

定义函数 $\Omega(\bar{n})$ 来表示在相反的方向映射：

$$\Omega(\bar{n}) = \{m \mid \xi(m) = \bar{n}\},\ m \in M,\ \bar{n} \in \bar{N} \tag{7}$$

定义 $q_{mp} \in \{0, 1\}$ 来表示虚拟化的网络功能的类型：

$$q_{mp} = \begin{cases} 1 \text{ if } m \text{ is of type } p \in P \\ 0 \text{ otherwise} \end{cases} \tag{8}$$

定义返回虚拟化的网络功能 m 类型的函数 $\tau(m)$.

$$\tau(m) = \{p \mid q_{mp} = 1\},\ m \in M,\ p \in P \tag{9}$$

如前所述，可以部署给定类型的虚拟化的网络功能一组特定的服务器，为了确保这一点，必须有：

$$q_{mp} = d\xi(m)p \tag{10}$$

虚拟化的网络功能只是代表可以提供特定类型的网络功能的地方，$y_m \in \{0, 1\}$ 表示虚拟化的网络功能是否有效

$$y_m = \begin{cases} 1 \text{ if } m \in M \text{ is active} \\ 0 \text{ otherwise} \end{cases} \tag{11}$$

3.1.2　添加虚拟交换机

接下来通过在每个虚拟化的网络功能和它所连接的原始交换机之间添加虚拟交换机来再次增加物理拓扑，该过程如图 1(c) 所示. 执行此步骤以简化网络流量守恒约束的表达式，这个过程不会增加解决方案空间的大小，因为仅考虑流动守恒约束.

3.2　线性建模

定义决策变量 x^t_{nm} 以表示业务节点到虚拟化的网络功能的映射：

$$x^t_{nm} = \begin{cases} 1 \text{ if node } n \in N^t \text{ is provisioned on m} \in \text{M} \\ 0 \text{ otherwise} \end{cases} \tag{12}$$

定义另一个变量来表示物理网络中负载节点和交换机之间的映射

$$z^t_{n\bar{s}} = \begin{cases} 1 \text{ if no de } n \in N^t \text{ is attached to switch } \bar{s} \\ 0 \text{ otherwise} \end{cases} \tag{13}$$

$z^t_{n\bar{s}}$ 不是决定变量，它可以从 x^t_{nm} 中导出

$$z^t_{n\bar{s}} = 1 \quad \text{if } x^t_{nm} = 1 \text{ and } \bar{h}_{\xi(m)\bar{s}} = 1 \tag{14}$$

因此，也可以从 x^t_{nm} 导出变量 y_m

$$y_m = 1 \quad \text{if } \sum_{t \in T} \sum_{n \in N^t} x^t_{nm} > 0 \tag{15}$$

假设$\hat{x}{}^{t}_{nm}$表示最后一个负载调配事件的x^{t}_{nm}的值，为了确保先前提供的负载的资源不被释放，必须具有$x^{t}_{nm} \geqslant \hat{x}{}^{t}_{nm}$，$\forall t \in \hat{T}$，$n \in N^t$，$m \in M$. 现在定义$\hat{y}_m \in \{0, 1\}$，表示最后一个负载的$y_m$值配置事件如下：

$$\hat{y}_m = 1 \quad \text{if} \sum_{t \in T} \sum_{n \in N^t} \hat{x}{}^{t}_{nm} > 0 \tag{16}$$

确保前面提供的负载资源不分配，必须满足$y_m \geqslant \hat{y}_m$，$\forall m \in M$. 接下来需要确保虚拟化的网络功能容量不会过高，活动虚拟化的网络功能的处理能力必须大于或等于通过总量，约束如下：

$$\sum_{t \in T} \sum_{n \in N^t} x^{t}_{nm} \times \beta' \leqslant c_{\tau(m)}, \ \forall m \in \{a \mid a \in M, y_a = 1\} \tag{17}$$

为了确保物理服务器容量的限制不被部署的虚拟化的网络功能影响，存在约束如下：

$$\sum_{m \in \Omega(n)} y_m \times k^{r}_{m} \leq c^{r}_{n}, \ \forall \bar{n} \in \bar{N}, r \in R \tag{18}$$

负载的每个节点必须映射到适当的虚拟化的网络功能类型，该约束表示如下：

$$x^{t}_{nm} \times \mathrm{g}^{t}_{np} = q_{mp}, \ \forall t \in T, n \in N^t, m \in M, p \in P \tag{19}$$

需要确保每个流量节点被配置，并且只需要一个虚拟化的网络功能.

$$\sum_{t \in T} \sum_{n \in N^t} x^{t}_{nm} = 1, \ \forall m \in M \tag{20}$$

定义的第二个决策变量来表示流量模型中的链路与物理网络中的链路之间的映射.

$$w^{tn_1n_2}_{\bar{u}\bar{v}} = \begin{cases} 1 \text{ if}(n_1, n_2) \in L^t \text{ uses physical link}(\bar{u}, \bar{v}) \\ 0 \text{ otherwise} \end{cases} \tag{21}$$

也假设$\hat{w}{}^{tn_1n_2}_{\bar{u}\bar{v}}$表示在最后一个流量配置事件$w^{tn_1n_2}_{\bar{u}\bar{v}}$的值，为了确保以前配置的流量的资源在当前的迭代中不被释放，定义如下约束

$$w^{tn_1n_2}_{\bar{u}\bar{v}} \geqslant \hat{w}{}^{tn_1n_2}_{\bar{u}\bar{v}}, \ \forall t \in \hat{T}, \ \forall (n_1, n_2) \in \{(a, b) \mid a \in N^t, b \in \eta^t(a), b > a\}, \ \forall \bar{u}, \bar{v} \in \bar{S} \tag{22}$$

为了确保流量请求中的每个定向链路都未映射到物理链路的两个方向，必须具有：

$$w^{tn_1n_2}_{\bar{u}\bar{v}} + w^{tn_1n_2}_{\bar{v}\bar{u}} \leqslant 1, \ \forall t \in T, \ \forall (n_1, n_2) \in \{(a, b) \mid a \in N^t, b \in \eta^t(a), b > a\}, \ \forall \bar{u}, \bar{v} \in \bar{S} \tag{23}$$

物理链路的容量约束：

$$\sum_{\bar{u} \in \bar{S}} \sum_{\bar{v} \in \bar{S}} (w^{tn_1n_2}_{\bar{u}\bar{v}} + w^{tn_1n_2}_{\bar{v}\bar{u}}) \times \beta' \leqslant \beta_{\bar{u}\bar{v}}, \ \forall t \in T, \ \forall (n_1, n_2) \in \{(a, b) \mid a \in N^t, b \in \eta^t(a), b > a\} \tag{24}$$

确保物理网络中每个交换机的负载在入口和出口交换机之外是相等的：

$$\sum_{\bar{v} \in \eta(\bar{u})} (w^{tn_1n_2}_{\bar{u}\bar{v}} - w^{tn_1n_2}_{\bar{v}\bar{u}}) = z^{t}_{n_1\bar{u}} - z^{t}_{n_2\bar{u}}, \ \forall t \in T, \ \forall (n_1, n_2) \in \{(a, b) \mid a \in N^t, b \in \eta^t(a), b > a\}, \ \forall \bar{u} \in \bar{S} \tag{25}$$

确保流量请求中的每个链接都在物理网络中的路径上进行配置：

$$\sum_{\bar{u} \in \bar{S}} \sum_{\bar{v} \in \bar{S}} (w^{tn_1n_2}_{\bar{u}\bar{v}} + w^{tn_1n_2}_{\bar{v}\bar{u}}) \geqslant 0, \ \forall t \in T, \ \forall (n_1, n_2) \in \{(a, b) \mid a \in N^t, b \in \eta^t(a), b > a\} \tag{26}$$

本文的目标是找到服务器的最佳数量和位置的最小化，再就是虚拟化运营成本和物理资源及网络中的碎片最小化.

虚拟化的网络功能的部署成本可以表示如下：

$$\mathbb{D} = \sum_{m \in M \mid y_m = 1} D^{+}_{p} \times q_{mp} \times (y_m - \hat{y}_m) \tag{27}$$

在不失一般性的情况下，假设服务器的能耗与正在使用的资源数量成正比，然而，即使在空闲状态下，服务器也通常会消耗电力，因此，计算服务器的功耗如下：

$$\mathbb{E}_{\bar{n}} = \sum_{m \in \Omega\bar{n}} y_m \times q_{mp} \times e^{r}(c^{r}_{n}, k^{r}_{p}) \tag{28}$$

$$e^{r}(r_t, r_c) = (e^{r}_{\max} - e^{r}_{\mathrm{idle}}) \times \frac{r_c}{r_t} + e^{r}_{\mathrm{idl}} \tag{29}$$

r_t和r_c分别表示总资源和消耗的资源. e^{r}_{idle}和$e^{r}_{\max}$分别表示资源r的空闲和峰值消耗状态中的能量成本.

因此，总能源成本是

$$\mathbb{E}=\sum_{\bar{n}\in\bar{N}}\sum_{m\in\Omega\bar{n}} y_m\times q_{mp}\times e^r(c^r_{\bar{n}},\ k^r_p) \tag{30}$$

负载转发成本：假设通过网络中的一个链路转发 1 Mbit 数据的成本是 σ(以美元计)，可以计算流量转发的总成本如下：

$$\mathbb{F}=\sum_{t\in T}\sum_{n_1\in N^t}\sum_{\substack{n_2\in\eta^t(n_1)\\ \text{and}n_2>n_1}}\sum_{\bar{u}\in\bar{S}}\sum_{\bar{v}\in\eta(\bar{u})}((w^{tn_1n_2}_{\bar{uv}}-\hat{w}{}^{tn_1n_2}_{\bar{uv}})\times\beta^t\times\sigma) \tag{31}$$

服务级目标违规处罚：可以计算实际流量经历的传播延迟如下：

$$\delta^a_t=\sum_{n_1\in N^t}\sum_{\substack{n_2\in\eta^t(n_1)\\ \text{and}n_2>n_1}}\sum_{\bar{u}\in\bar{S}}\sum_{\bar{v}\in\eta(\bar{u})} w^{tn_1n_2}_{\bar{uv}}\,\delta_{\bar{uv}} \tag{32}$$

令 $\rho^t(\omega^t,\ \delta^t,\ \delta^t_a)$ 作为计算服务级目标违规惩罚的函数，给定了判定罚分(ωt)的政策，预期传播延迟(δ^t)和实际传播延迟(δ^t_a)，所以违反服务级目标违规的总成本可以表示如下：

$$\mathbb{P}=\sum_{t\in T}\rho^t(\omega^t,\ \delta^t,\ \delta^t_a) \tag{33}$$

3.3　碎片代价

第二个目标是尽量减少活动服务器和链接的资源(服务器和链路)碎片，使用与上述费用相同的单位来表达. 为此，假设 p^r 表示 $r\in R$ 型单位资源的价格，也将 ρ^β 表示为单位带宽的价格. 如果物理服务器至少有一个活动的虚拟化的网络功能，则该物理服务器被认为是活动的，二进制变量 $a_{\bar{n}}$ 捕获此属性：

$$a_{\bar{n}}=\begin{cases}1\ \text{if}\ \sum_{m\in\Omega(\bar{n})} y_m>0\\ 0\ \text{otherwise}\end{cases} \tag{34}$$

类似地，物理链路($\bar{u}$，$\bar{v}$)被认为是活动的，如果它承载至少一个业务流，使用二进制变量 $f_{\bar{uv}}$ 表示：

$$f_{\bar{uv}}=\begin{cases}1\ \text{if}\ \sum_{t\in T,\ (n_1,\ n_2)\in L^t} w^{tn_1n_2}_{\bar{uv}}>0\\ 0\ \text{otherwise}\end{cases} \tag{35}$$

资源碎片的总成本如下

$$\begin{aligned}\mathbb{C}=&\sum_{\bar{n}\in\bar{N}} a_{\bar{n}}\sum_{r\in R}(c^r_{\bar{n}}-\sum_{m\in\Omega(n)}(k^r_p\times q_{mp}\ y_m))\ p^r+\\ &\sum_{\bar{u}\in\bar{S}}\sum_{\bar{v}\in\eta(\bar{u})} f_{\bar{uv}}(\beta_{\bar{uv}}-\sum_{t\in T}\sum_{n_1\in N^t}\sum_{\substack{n_2\in\eta^t(n_1)\\ \text{and}n_2>n_1}}(w^{tn_1n_2}_{\bar{uv}}\times\beta^t))\ \rho^\beta\end{aligned} \tag{36}$$

第一项表示服务器资源分片的成本(CPU、存储器、磁盘等)，第二项表示链路带宽分段的成本，目标是最小化网络运营成本和资源分割的总和，这可以表示为上述成本的加权和.

$$minimize(\alpha\ \mathbb{D}+\beta\ \mathbb{E}+\gamma\ \mathbb{F}+\lambda\ \mathbb{P}+\mu\ \mathbb{C}) \tag{37}$$

α，β，γ，λ，μ 是用于调整成本组件的相对重要性的加权因子.

4　启发式算法求解

(37)式的求解是一个 NP 难问题. 将虚拟化的网络功能编排转化为单源限制容量工厂位置问题的 NP 难问题[17]. 在文献[17]中，给出了一套具有固定成本和容量的生产工厂的潜在位置，由这些工厂生产的商品将提供给具有固定需求和相关运输成本的一组客户. 此外，每个客户必须由一个工厂服务，目标是找到应该运行的平台的小部分，以尽量减少成本，而不会违反能力和需求限制. 考虑到文献[17]的一个实例，可以通过以下方式将其转换为虚拟化的网络功能编排的实例：①对于每个客户，创建链→工厂→客户，其中创建链是虚拟入口交换机，客户是出口交换机，工厂是虚拟化的网络功能；②将链路的带宽设置为等于客户需求；③使用运输成本作为流量转发成本；④配置每台物理机器部署单个虚拟化的网络功能的工厂；⑤将每个工厂的加工能力设定为等于其生产能力. 这些操作可以在问题大小的多项式时间内执行. 现在，如果可以解决这个虚拟化的网络功能编排实例，也会得到一个文献[17]的解决方案. 然而，文献[17]是

NP-hard，虚拟化的网络功能编排也是NP-hard.

在本节中，提出一个启发式算法来解决虚拟化的网络功能编排. 给定网络拓扑，一组中间件规范和一批流量请求，启发式查找以最小营运成本部署网络所需的不同类型的虚拟化的网络功能的数量和位置，没有指明用简单而快速的启发式算法来处理资源碎片的问题. 然而，实验结果表明，即使这种简化，启发式产生非常接近最优的解决方案. 启发式运行两个步骤，首先，将虚拟化的网络功能编排建模为具有相关成本的多阶定向图. 然后，通过运行算法，从多级图中找到一个接近最优的虚拟化的网络功能位置[18].

4.1 多阶段图形建模

对于给定的负载请求$t=\langle \bar{u}^t, \bar{v}^t, \psi^t, \beta^t, \delta^t, \omega^t\rangle$，用$l_{\psi^t}+2$阶段将$t$表示为多阶段图. 起始和终端阶段(*i.e.*，$l_{\psi^t}+2$)代表入口和出口交换机，这两个阶段只包含一个节点$\bar{u}^t$，$\bar{v}^t$. 任意阶段$i(i\in\{2,\cdots(l_{\psi^t}+1)\})$表示业务请求中的第$(i-1)$个虚拟化的网络功能，并且该阶段内的节点表示可以放置该类型的虚拟化的网络功能的可能的服务器位置，每个节点的部署与虚拟化的网络功能部署成本和能量成本相关联.

这个多阶图中的边$(\bar{v}_i, \bar{v}_j)$表示虚拟化的网络功能在连接到交换机$\bar{v}_j$的服务器上的位置，假设序列中的虚拟化的网络功能部署在连接到交换机$\bar{v}_i$的服务器上，在i和$i+1$阶段的所有节点对之间设置一个有向边，将两个成本与每个边缘相关联：转发流量的成本(31)式和违反服务级目标违规的惩罚(33)式，流量转发成本与交换机之间的加权最短路径(以延迟为单位)成比例，通过以下过程获得对服务级目标违规行为的处罚：

1) 同样划分阶段之间的最大允许延迟；

2) 为多阶段图表中的两个连续阶段之间的转换分配服务级目标违规成本，每当导致由于节点处的流量传输和处理而导致的分配延迟.

通过对(37)式之后的节点和边缘成本相加来计算两个连续阶段之间的转换的总成本. 最后，从第一阶段的节点到最后阶段的节点的路径代表了虚拟化的网络功能的放置，我们的目标是在多阶段图中找到产生最小营运成本的路径.

4.2 启发式算法

算法1给出了启发式解决方案的伪码，将负载请求t作为输入，并以每个交换机的资源容量注释网络拓扑图$\bar{G}$，保留成本和π两个表，以跟踪中间件展示位置的成本和顺序. $cost_{i,j}$表示在中间件序列ψ^t中将第j个中间件部署到连接有交换机i的服务器的成本，使用一些帮助程序来实现. 第一个帮助程序“IsResourceAvailable”检查是否可以在交换机i上放置用于流量请求t的中间件$mbox$，以满足最低带宽和资源要求. 第二个帮助者*GetCost*计算在连接到交换机j的服务器上为流量请求t放置中间件$mbox$的成本. 在考虑到j时产生当前节点的最小成本，先前节点k由条目$\pi_{k,j}$跟踪，最后，使用π中的条目来追溯以获得所需的中间件序列.

Algorithm 1 *ProvisionTraffic*$(t, \bar{G})$

1: $\forall(i,j)\in\{1\ldots|\psi^t|\}\times\{1\ldots|\bar{S}|\}$: $cost_{i,j}\leftarrow\infty$, $\pi_{i,j}\leftarrow NIL$

2: $Vi\in|\bar{S}|$:

3: if *IsResourceAvailable*(u^t, i, Ψ_1^t, t) then

4: $cost_{1,n}\leftarrow GetCost(u^t, i, \Psi_1^t, t)$, $\pi_{1,n}\leftarrow n$

5: end if

6: $\forall(i,j,k)\in\{2\ldots|\Psi^t|\}\times\{1\ldots|\bar{S}|\}\times\{1\ldots|\bar{S}|\}$:

7: if *IsResourceAvailable*(k, j, Ψ_i^t, t) then

8: $cost_{i,j}\leftarrow\min\{cost_{i,j}, cost_{i-1,k}+GetCost(k, j, \Psi_i^t, t)\}$

9: $\pi_{i,j}\leftarrow i$ yielding minimum $cost_{i,j}$

10: end if

11: $\prod\leftarrow NIL$, $C\leftarrow\infty$, $\psi\leftarrow\langle\rangle$

12: $\forall i\in|\bar{S}|$:

13：　$C \leftarrow \min\{C, cost_{|\Psi^t|,i} + ForwardingCost(i, v^t) + SLOViolation\ Cost(i, v^t, t)\}$

14：　$\prod \leftarrow i$ yielding minimum $cost_{|\Psi^t|,i}$

15：$\forall i \in \langle |\Psi^t|, |\Psi^t| - 1 \ldots 1 \rangle$：Append $\prod$ to ϕ，$\prod \leftarrow \pi_{i,\Pi}$

16：return $Reverse(\phi)$

5　仿真实验

拓扑数据集：我们把虚拟化的小范围网络称为 Internet2. ①Internet2 研究网络(12 个节点，15 个链路)，②大学数据中心网络(23 个节点，42 个链路)，③Rocketfuel 拓扑数据集(79 个节点，147 个链路)的自治系统 3967(AS-3967)[19].

业务数据集：使用实际跟踪和综合生成的流量进行评估. 使用流量矩阵跟踪 Internet2 拓扑生成的时变流量，该跟踪包含 12×12 流量矩阵的快照并显示流量的变化. 最后，对于 Rocketfuel 拓扑结构，使用 FNSS 工具生成了一个合成的时变流量矩阵[20]，它遵循文献[19]的分布并显示时间效应.

中间件和成本数据：根据文献[4]和文献[8]中提供的数据，为每个流量生成了一个长度为 3 的中间件序列. 已经使用制造商和服务提供商的公开数据表来选择和推断服务器能源成本，服务级目标违规成本(违反最大延迟时间)，软件资源需求中间件及其处理能力，还从流行的网络设备制造商那里获得了硬件中间件的能耗数据.

虚拟化的网络功能和硬件中间件放置位置的拓扑特性如图 2－图 4 所示，入口交换机和中间件之间跳数 *CDF* 如图 2 所示. 与硬件中间件相比，80%的虚拟化的网络功能位于入口交换机的 2 跳内(大多数为 1 跳)，只有 4%的虚拟化的网络功能位于 4 跳距离，这说明将虚拟化的网络功能放置更远的地方，可以通过降低能源成本来减少营运成本.

对于中间件和出口开关之间的跳跃距离获得类似的结果(图 3). 这两个数字也表明了 CPLEX 在入口和出口交换机之间的路径上以更平衡(对称)的方式放置中间件.

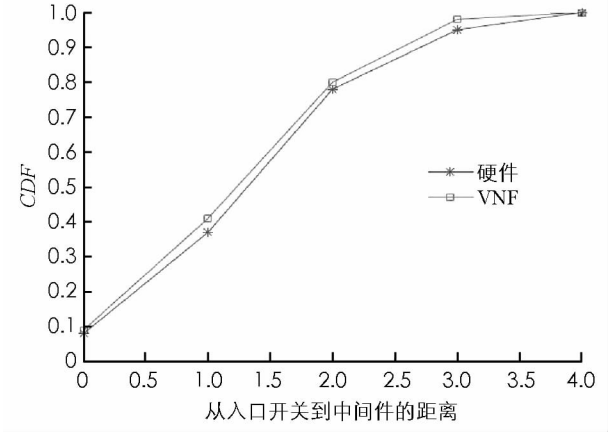

图 2　入口开关到中间件的距离

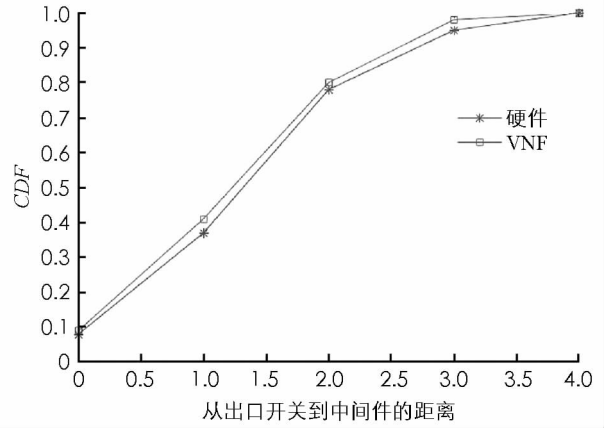

图 3　出口开关到中间件的距离

硬件中间件和虚拟化的网络功能的路径迁移性能如图 4 所示. 虚拟化的网络功能比硬件中间件具有更高的迁移性，因为虚拟化的网络功能位置不像硬件中间件固定，它们可以在任何服务器上进行配置以减少营运成本.

图 5 显示了 Internet2 拓扑的平均利用率，VNF 的平均利用率高于硬件中间件的平均利用率，VNF 中间件在更高的流量期间通过部署更多的虚拟化的网络功能来降低营运成本，VNF 中间件的代价是使用了更多的服务器，实现了资源利用率是硬件中间件的 1.1 倍.

图 6 显示了数据中心网络的服务器资源利用相关的结果，在数据中心网络的情况下，VNF 的利用率也更高.

Internet2 和数据中心网络的中间件部署的拓扑特性如图 7－图 9 所示，图 7 显示了从入口交换机到虚拟化的网络功能中间件的跳线距离的 *CDF*，数据中心网络的 VNF 的跳跃距离非常接近于最优解，在数据

中心网络的情况下，存在较大的差距，这是由于数据中心网络提供的更高的路径分集，每对节点具有多于一个相等的成本路径. 数据中心网络的 VNF 通过探索所有这些解决方案找到最佳，每对节点具有多于一个相等的成本路径. 然而，启发式总是选择第一条最短路径，它没有探索将执行时间保持在实际限制之内的备选路径.

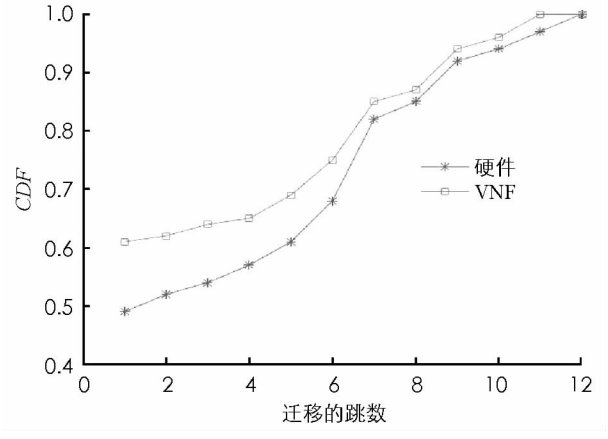

图 4 中间件的迁移性比较

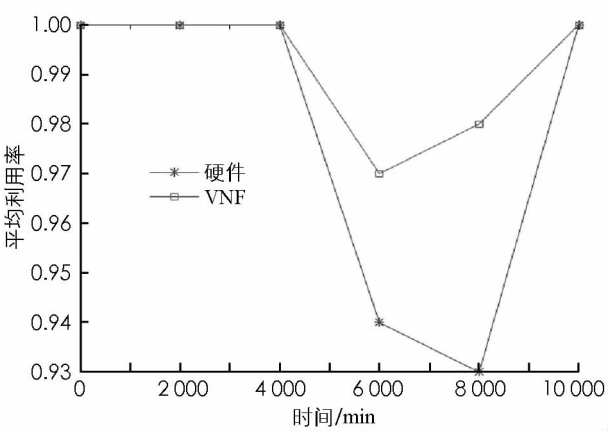

图 5 Internet2 拓扑的资源利用率的比较

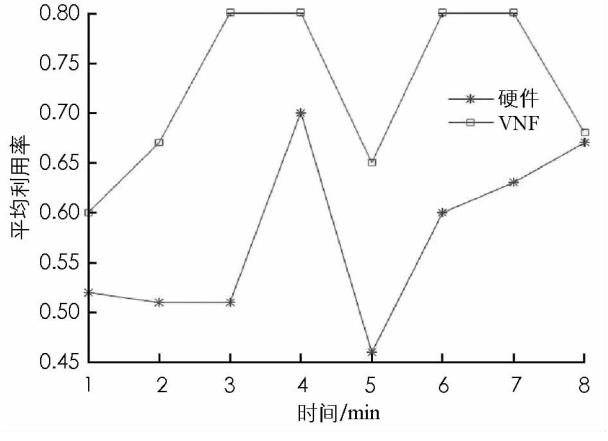

图 6 数据中心资源利用率的比较

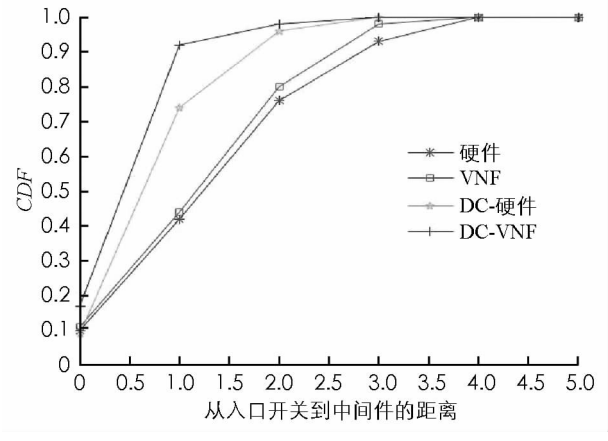

图 7 入口开关到中间件的距离的 *CDF* 比较

对于出口情况，观察到类似的结果(图 8). 从图 7 和图 8 可以看出，*CDF* 非常相似，这意味着 VNF 均匀地放置在入口和出口交换机之间的路径上.

路径迁移如图 9 所示. 如前所述，数据中心网络的 VNF 的性能接近于最优解，在数据中心网络的情况下，启发式方法具有较大的迁移性.

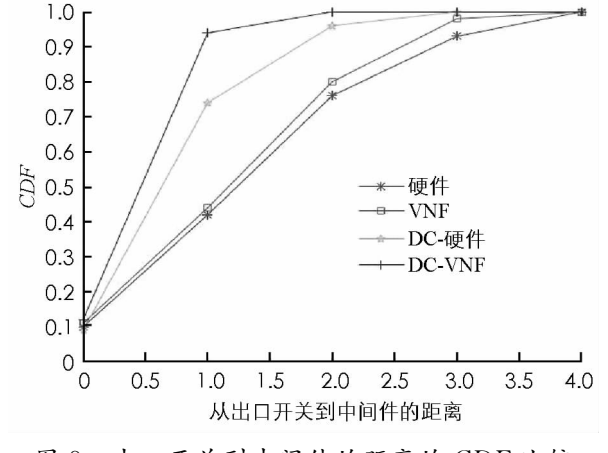

图 8 出口开关到中间件的距离的 *CDF* 比较

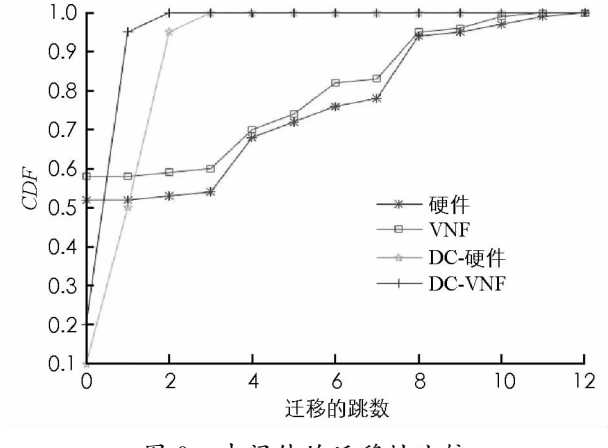

图 9 中间件的迁移性比较

6 结　论

在本文中，提出了虚拟化的网络功能编排问题的两个解决方案：小型网络最佳解决方案和大型网络(数据中心)的启发式算法. 仿真结果表明，所提出的启发式算法可以降低网络运营成本，相关性能优于传统的硬件中间件方法. 下一步工作的重点是通过部署备份虚拟化的网络功能来提高故障恢复能力，并加强现有算法性能，进一步减少最优解的运行时间.

参考文献：

[1] SEKAR V, RATNASAMY S, REITER M K, et al. The Middlebox Manifesto [C]//Proceedings of the 10th ACM Workshop on Hot Topics in Networks-HotNets'11, Cambridge, Massachusetts, November 14-15, 2011.

[2] SHERRY J, HASAN S, SCOTT C, et al. Makingmiddleboxes Someone Else's Problem: Network Processing As a Cloud Service [J]. Acm Sigcomm Computer Communication Review, 2012, 42(4): 13-24.

[3] JOSEPH D A, TAVAKOLI A, STOICA I. A Policy-Aware Switching Layer for Data Centers [J]. Acm Sigcomm Computer Communication Review, 2008, 38(4): 51-62.

[4] QAZI Z A, RUI M, TU, et al. SIMPLE-Fying Middlebox Policy Enforcement Using SDN [J]. Acm Sigcomm Computer Communication Review, 2013, 43(4): 27-38.

[5] QUINN P, GUICHARD J. Service Function Chaining: Creating a Service Plane via Network Service Headers [J]. Computer, 2014, 47(11): 38-44.

[6] GADRE A, ANBIAH A, SIVALINGAM K M. Centralized Approaches for Virtual Network Function Placement in SDN-Enabled Networks [J]. Eurasip Journal on Wireless Communications & Networking, 2018, 2018(1): 197.

[7] XU Q, GAO D Y, LI T X, et al. Low Latency Security Function Chain Embedding across Multiple Domains [J]. IEEE Access, 2018, 6: 14474-14484.

[8] KARL H, DRÄXLER S, PEUSTER M, et al. DevOps for Network Function Virtualisation: an Architectural Approach [J]. Transactions on Emerging Telecommunications Technologies, 2016, 27(9): 1206-1215.

[9] MIJUMBI R, SERRAT J, GORRICHO J L, et al. Network Function Virtualization: State-of-the-art and Research Challenges [J]. IEEE Communications Surveys & Tutorials, 2016, 18(1): 236-262.

[10] MARTINS J, AHMED M, RAICIU C, et al. ClickOS and the Art of Network Function Virtualization [C]//Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation, 2014: 459-473.

[11] 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.

[12] GEMBER-JACOBSON A, VISWANATHAN R, PRAKASH C, et al. OpenNF: Enabling in Novation in Network Function Control [C]// Acm Conference on Sigcomm. ACM, 2014.

[13] MEHRAGHDAM S, KELLER M, KARL H. Specifying and Placing Chains of Virtual Network Functions [C]//2014 IEEE 3rd International Conference on Cloud Networking (CloudNet), Luxembourg, Luxembourg, October 8-10, 2014.

[14] COHEN R, LEWIN-EYTAN L, NAOR J S, et al. Near Optimal Placement of Virtual Network Functions [C]//2015 IEEE Conference on Computer Communications (INFOCOM), Kowloon, Hong Kong, China, April 26-May 1, 2015.

[15] LUO D X, XU H T, ZHEN Y, et al. Learning Mixtures of Markov Chains from Aggregate Data with Structural Constraints (extended Abstract) [C]//2017 IEEE 33rd International Conference on Data Engineering (ICDE), San Diego, CA, USA, April 19-22, 2017.

[16] CHOWDHURY N M M K, BOUTABA R. A Survey of Network Virtualization [J]. Computer Networks, 2010, 54(5): 862-876.

[17] SRIDHARAN R. A Lagrangian Heuristic for the Capacitated Plant Location Problem with Side Constraints [J]. Journal of the Operational Research Society, 1991, 42(7): 579-585.

[18] FORNEY G D. Theviterbi Algorithm [J]. Proc IEEE, 1993, 61(5): 268-278.

[19] NGUYEN H X, THIRAN P. Active Measurement for Multiple Link Failures Diagnosis in IP Networks [M]//Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004: 185-194. DOI: 10.1007/978-3-540-24668-8_19.

[20] SAINO L, COCORA C, PAVLOU G. A Toolchain for Simplifying Network Simulation Setup [C]//Proceedings of the Sixth International Conference on Simulation Tools and Techniques, Cannes, France, March 5-7, 2013.

On Algorithm of Functions Dynamic Orchestration in Virtualized Network

MU Ze-ping

School of Artificial Intelligence & Big Data, Chongqing College of Electronic Engineering, Chongqing 401331, China

Abstract: With the development of informatization, network businesses are expanding and business functions are becoming more powerful; thus, network infrastructure has begun to face new challenges in serving businesses. Research regarding the dynamic deployment of network slices according to business requirements is urgently needed. Without violating the service level protocol, the virtualized network function orchestration problem has been studied, and an integer linear programming mathematical model in virtualized network been put forward. Then, heuristic algorithm based on dynamic orchestration is used to solve the model. Finally, the algorithm is tracked and simulated in the real world network topology. The results show that the proposed heuristic algorithm can reduce the network operation cost and achieve better performance than the traditional hardware middleware method.

Key words: network function virtualization; network function orchestration; heuristic algorithm

责任编辑 汤振金