



基于在线实例配置的服务功能链部署方法

孙士清, 彭建华, 游 伟, 李英乐

(信息工程大学 信息技术研究所, 郑州 450001)

摘 要: 为应对实际环境中网络流量的动态变化, 同时降低运营商的运营成本, 提出基于在线服务功能扩展的服务功能链部署方法。将空置状态虚拟功能实例的生命周期管理问题建模为雪橇租赁模型, 采用雪橇租赁问题的最优解配置空置服务功能实例的生命周期, 并设计自适应扩展开销最小化的在线服务功能实例扩展算法。以服务功能实例自适应扩展的结果作为约束条件, 将每一时刻的服务功能链部署问题建模为整数线性规划问题, 利用遗传算法求解带宽开销最小化的服务功能路径, 实现动态场景下的服务功能链部署。仿真结果表明, 该方法能根据网络流量变化动态调整虚拟资源, 降低服务功能实例自适应扩展成本, 节省带宽资源。

关键词: 服务功能链; 软件定义网络; 网络功能虚拟化; 长短时记忆; 遗传算法

开放科学(资源服务)标志码(OSID):



中文引用格式: 孙士清, 彭建华, 游伟, 等. 基于在线实例配置的服务功能链部署方法[J]. 计算机工程, 2019, 45(12): 71-78.

英文引用格式: SUN Shiqing, PENG Jianhua, YOU Wei, et al. Service function chain deployment method based on online instance configuration[J]. Computer Engineering, 2019, 45(12): 71-78.

Service Function Chain Deployment Method Based on Online Instance Configuration

SUN Shiqing, PENG Jianhua, YOU Wei, LI Yingle

(Research Institute of Information Technology, Information Engineering University, Zhengzhou 450001, China)

[Abstract] To cope with the dynamic changes of network traffic in the actual environment and reduce the running cost of operators, this paper proposes a Service Function Chain (SFC) deployment method based on online service function extension. The method firstly simplifies the life cycle management of virtual function instances in idle state to a ski-rental problem, and adopts the optimal solution of the ski-rental problem to configure the life cycle of service function instances in idle state. So an online service function instance extension algorithm that adaptively minimizes extension costs is designed. Then, with adaptive extension result of service function instances as constraints, the SFC deployment at each moment is modeled as a integer linear programming problem. The Genetic Algorithm (GA) is used to find the service function path with minimized bandwidth costs, so as to implement SFC deployment in dynamic scenarios. Simulation results show that the proposed method can dynamically adjust virtual resources based on network traffic changes, reducing bandwidth occupancy and adaptive extension costs of service function instances.

[Key words] Service Function Chain (SFC); Software Defined Network (SDN); Network Function Virtualization (NFV); Long Short-Term Memory (LSTM); Genetic Algorithm (GA)

DOI: 10.19678/j.issn.1000-3428.0055489

0 概述

在运营商网络中, 流量需要经过不同的虚拟网络功能 (Virtual Network Function, VNF) 处理, 实现不

同的网络服务^[1]。网络功能也称为服务功能 (Service Function, SF), 一组有序的服务功能列表构成了服务功能链 (Service Function Chain, SFC)^[2]。传统的网络功能以“网络应用”或者“中间件”的形式进行部署, 其

基金项目: 国家重点研发计划 (2016YFB0801605); 国家自然科学基金创新研究群体项目 (61521003); 国家自然科学基金 (61801515)。

作者简介: 孙士清 (1992—), 男, 硕士研究生, 主研方向为网络功能虚拟化; 彭建华, 教授; 游 伟, 助理研究员、博士; 李英乐, 助理研究员、硕士。

收稿日期: 2019-07-15 **修回日期:** 2019-08-26 **E-mail:** 976355253@qq.com

中软件和硬件紧密耦合,并且由特定的设备运营商统一提供^[3]。基于专有硬件的中间件实现使得网络僵化、可扩展性差:一方面,运营商通常按照峰值流量进行网络建设,资源使用率较低;另一方面,设计、部署以及运营端到端的网络服务通常通过手动配置和管理服务功能链,大量的网络功能设备导致在人工配置和管理过程中极易出错,新服务的上线时间长^[4]。软件定义网络(Software Defined Network, SDN)^[5]和网络功能虚拟化(Network Function Virtualization, NFV)^[1]的出现,使得运营商可以动态创建 SFC 并执行相应的流量转发策略,实现网络服务的灵活部署。SDN 将控制平面和数据平面进行分离,赋予了网络可编程能力,允许控制平面对 SFC 路径转发规则进行编程,并下发到 SDN 交换机上。NFV 通过虚拟化技术将服务功能与硬件进行解耦,服务功能软件转化为虚拟网络功能并运行在通用服务器上,实现了服务功能实例的动态部署和虚拟链路的动态创建^[6-7]。

目前,SFC 部署方法已经成为学术研究的热点问题。文献[8]从运营成本优化的角度研究 SFC 部署问题,设计基于贪婪思想的启发式算法和遗传算法(Genetic Algorithm, GA)的 VNF 部署方法,减少服务功能实例和活跃物理节点的数量,以降低运营开销。文献[9]提出基于 Q-learning 算法的 SFC 部署方法,该方法首先对 SFC 部署问题构建了 0-1 规划模型,然后将不同部署方案的选取建立为马尔科夫决策模型,采用 Q-学习算法求解最优 SFC 部署方案。文献[8-9]的工作专注于离线 SFC 部署问题,忽略了流量的动态变化,难以实现虚拟资源使用的动态管理。文献[10]提出在线 VNF 配置方法,但是未对该场景下服务功能链部署问题进行研究。文献[11]提出基于业务请求单元缓存负载预测的 SFC 动态部署方法,该方法首先采用长短时记忆(Long Short-Term Memory, LSTM)神经网络预测未来缓存负载状态,然后基于预测结果调整虚拟节点上的 VNF 计算资源分配,并重新计算 SFC 的服务功能路径,以降低 SFC 端到端时延,但该方法没有考虑离线训练的神经网络模型不能适应网络流量模式可能发生的变化,并且每个时间段对所有 SFC 路由路径进行调整会造成网络抖动,增加服务等待时间,降低网络服务质量。同时,文献[12]的实验结果表明,基于 RNN、LSTM 等传统深度神经网络的时间序列方法在网络流量数据集上的准确率仅为 40%,该方法没有考虑在难以获得精确预测结果的情况下服务提供的持续性和稳定性问题。文献[13]采用 FTRL 算法预测每条 SFC 的网络流量,当未来时刻所需服务功能实例数量大于当前在用实例数量时,部署更多的新的服务功能实例,这种方式不能充分利用空闲的服务功能实例,造成虚拟资源浪费。

本文面向网络流量难以预测的场景,提出在线服务功能实例配置算法,根据服务功能请求的动态变化配置服务功能实例的数量,并基于雪橇租赁模型设置空闲服务功能实例的生命周期,实现虚拟资源使用的动态管理,提高底层资源使用效率,降低运营开销。基于此,给出服务功能路径计算方法,为每一个 SFC 请求计算路由路径,实现动态场景下的服务功能链部署。

1 SFC 动态体系结构与模型建立

1.1 SFC 体系结构

根据 IETF R7665^[2] 参考文件,SFC 体系结构如图 1 所示,由编排平面、控制平面和数据平面三部分构成。编排平面由多种网络功能应用构成,用户可以通过仪表模板、命令行或者调用应用程序接口(Application Program Interface, API)等方式创建服务功能链并进行部署,无需关注具体的实现细节。控制平面由策略控制器、SDN 控制器和 NFV 管理与编排(NFV Management and Orchestration, NFV MANO)三部分构成。策略控制器负责将用户指令转化为具体的流分类策略(不同数据流对应的服务功能集合和访问顺序等),并将其下发至服务流分类器;SDN 控制器根据流分类策略制定不同数据流的服务功能路径(Service Function Path, SFP),并将其下发至服务功能转发器(Service Function Forwarder, SFF),实现数据流量的灵活调度;NFV MANO 负责对底层资源进行管理和编排,实现对服务功能实例的创建、状态监控、扩展以及删除等全生命周期管理^[14-15]。数据平面由服务流分类器、服务功能转发器和服务功能三部分构成,其中流分类器执行策略控制器下发的流分类策略,对到达的数据流进行分类,并指定相应的服务功能路径,SFF 使用数据包中的 SFC 头部信息将数据包转发至与其关联的一个或者多个 SF;SF 负责对接收到的数据流量进行特定的功能性处理,在虚拟化的网络环境中,其一般由虚拟组件(如 VNF)进行实现,并部署在物理机上。一个 SF 实例可以接收一个或者多个 SFF 的数据流量。

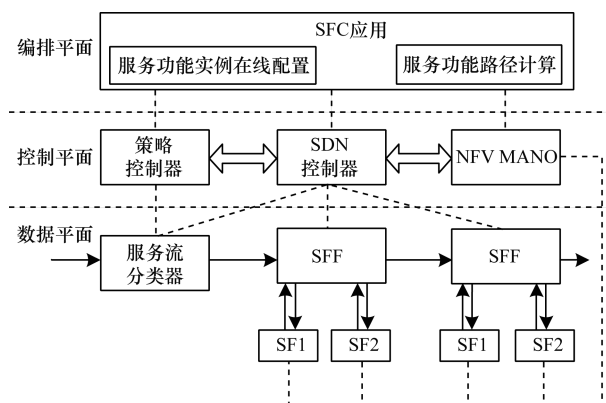


图 1 SFC 体系结构

除此之外,参考文件中还定义了 SFC 域的概念,指实现 SFC 的网络或者网络区域。一个 SFC 域仅限于单个网络管理域,本文基于单个 SFC 域,在编排平面设计在线服务功能实例配置、服务功能路径计算算法,实现 SFC 部署和服务功能实例的动态管理。

1.2 SFC 在线部署模型

1.2.1 网络模型

将网络拓扑抽象为赋权无向图 $G(V, E)$, 其中, V 代表底层节点集合, E 代表底层链路集合。每个底层节点代表一个计算节点, 由于基于软件的网络应用大多是 CPU 密集型, 一般情况下不会占用大量的存储和内存资源^[16], 因此仅考虑其计算资源, 以能够虚拟出的 vCPU 数量进行表示, 记为 C_v 。每条链路 $(u, v) \in E$, 带宽记为 B_{uv} , 时延记为 L_{uv} 。在基于 NFV 的网络环境中, 每个虚拟机仅实例化一个 VNF, 对应于一个服务功能实例, 服务功能类型用集合 $F = \{1, 2, \dots, f\}$ 进行表示, 类型为 f 的服务功能吞吐量为 ct_f , 所需 vCPU 数量为 cr_f , $f^k \in f$ 表示类型为 f 的服务功能的一个具体实例。服务功能实例可以设置活跃(active)和空置(idle)两种状态, 对设置 idle 状态的 VNF 实例, 设置其生命周期 deadline, 并不再向其映射新的 SFC 请求。在 t 时刻系统中的 active 状态的 f 类型 VNF 实例集合为 $Z_f(t)$, 数量为 $z_f(t)$, 相应地, idle 状态的 VNF 实例集合为 $W_f(t)$, 数量为 $w_f(t)$ 。

1.2.2 SFC 请求模型

在任意时刻, 网络环境中的 SFC 请求集合为 $S(t)$, 每个 SFC 请求 s_i 可以用六元组 $\langle \alpha_i, \beta_i, G_i, b_i, l_i, \Delta t_i \rangle$ 进行表示。其中, α_i 和 β_i 分别为服务功能链流量的流入节点和流出节点, G_i 为服务功能链 s_i 中流量路由经过的服务功能集合, 每个元素 g_{ij} 代表流量经由的第 j 个服务功能, 参量 $g_{ij}^{f^k} = 1$ 时代表服务功能链 s_i 的第 j 个服务功能为 f^k , 否则为 0。特别地, 对于流量流入节点, 记 $g_{i0} = \alpha_i$, $g_{i,0}^{f^k} = 0$, 对于流量流出节点, 记 $g_{i, |G_i|+1}^{f^k} = 0$, b_i 为请求所需流量带宽, l_i 为时延阈值, Δt_i 为该请求在系统中的驻留时间。

1.2.3 优化目标

本文建立了动态 SFC 架构下的自适应扩展开销最小化模型。将网络系统运营的时间划分为离散时刻 $\Gamma = \{0, 1, \dots\}$, 每个时间间隔为 δ 。运营商需要根据 SFC 请求的动态变化部署或撤销服务功能实例, 以在满足服务需求的情况下减少运营开销。在本文提出的模型中, 将当前时刻提供正常服务所必需的服务功能实例状态设置为 active, 其他服务功能实例设置为 idle 状态, 运营商需要通过网络流量的变化动态调整服务功能实例状态, 自适应扩展提供网络服务所需资源。自适应扩展开销可分为:

1) idle 状态服务功能实例运行开销: 保留 idle 状态的服务功能实例会增加能耗开销和虚拟资源占用开销。设 ϕ_f 为单个时间间隔 δ 内类型 f 的服务功能实例运行开销系数, 则在系统运行时间段内, 服务功能实例运行开销为:

$$O = \sum_{t \in \Gamma} \sum_{f \in F} \phi_f w_f(t) \quad (1)$$

2) 服务功能实例部署开销: 部署新的服务功能实例需要进行虚拟机的镜像传输并增加服务的等待时延, 产生部署开销。设 φ_f 为类型 f 的服务功能实例的部署开销系数, $d_f(t)$ 为 t 时隙部署的类型 f 的服务功能实例数量, 则在整个系统运行时间段内, 服务功能实例的部署开销为:

$$D = \sum_{t \in \Gamma} \sum_{f \in F} \varphi_f d_f(t) \quad (2)$$

整个系统在运行时间内的优化目标为:

$$\min(O + D) \quad (3)$$

在每一时刻对服务功能实例进行动态调整后, 需要对到达的 SFC 请求配置服务功能路径。本文以当前时刻 active 状态服务功能实例数量和部署情况作为约束条件, 以当前时刻带宽开销最小化为目标部署新到达的 SFC 请求, 优化目标为:

$$\min \sum_{u \in V} \sum_{s_i \in S(t)} \sum_{j=0}^{|G_i|} y_{uv}^{ij} b_i \quad (4)$$

其中, 二值变量 y_{uv}^{ij} 值为 1 时表示 g_{ij} 与 $g_{i,j+1}$ 之间的路径先后经过相邻的两个节点 u, v , 否则为 0。

1.2.4 SFC 请求部署约束

定义二值变量 $x_v^{f^k}$, 其值为 1 时表示服务功能实例 f^k 部署在节点 v 上, 否则为 0。在任意时刻, 系统中的 SFC 部署需要满足以下约束:

1) 对于已经成功部署在底层网络的 SFC, 式(5)确保其所需的服务功能都能映射到底层节点的服务功能实例上, 且每个服务功能仅映射到一个服务功能实例上。

$$\sum_{v \in V} g_{ij}^{f^k} x_v^{f^k} = 1, \forall s_i \in S(t), j \in G_i, f^k \in Z_f(t), f \in F \quad (5)$$

2) 对于每个节点来说, 式(6)表示对每个 VNF 实例 f^k 处理的数据流量不能超过其吞吐量, 式(7)表示在节点 v 上的 VNF 实例化不能超过该节点的计算资源总量。

$$\sum_{s_i \in S(t)} \sum_{g_{ij} \in G_i} g_{ij}^{f^k} b_i \leq ct_f, \forall f^k \in Z_f(t), f \in F \quad (6)$$

$$\sum_{f \in F} \sum_{f^k \in Z_f(t)} x_v^{f^k} cr_f \leq C_v, \forall v \in V \quad (7)$$

对于每一条物理链路, 式(8)表示其上的 SFC 数据流量不能超过该链路的流量带宽。对于每个 SFC 数据流量, 式(9)表示其仅能通过一条路径在相邻服务功能之间路由, 同时两个服务功能的路由路径不能出现环路。式(10)表示 SFC 流量路径中的流量守恒。

$$\sum_{s_i \in S(t)} \sum_{j=0}^{|G_i|} y_{uv}^{ij} b_i \leq B_{uv}, \forall (u, v) \in E \quad (8)$$

$$\sum_{u \in V} y_{vu}^{ij} \leq 1, \sum_{u \in V} y_{uv}^{ij} \leq 1 \quad (9)$$

$$\forall s_i \in S(t), j \in [0, |G_i|]_Z, v \in V \quad (9)$$

$$\sum_{u \in V} (y_{vu}^{ij} - y_{uv}^{ij}) = g_{ij}^{f^k} x_v^{f^k} - g_{i,j+1}^{f^k} x_v^{f^k} \quad (10)$$

$$\forall s_i \in S(t), j \in [0, |G_i|]_Z, f, f' \in Z_f(t), v \in V \quad (10)$$

3) 需要考虑底层链路时延对 SFC 造成的影响, 式(11)表示 SFC 请求部署不能超过其时延要求。

$$\sum_{g_{ij} \in G_i} \sum_{(u,v) \in E} y_{uv}^{ij} L_{uv} \leq l_i, \forall s_i \in S(t) \quad (11)$$

2 算法设计

在线服务功能链部署算法首先在每个时间间隙 $t \in \Gamma$, 根据该间隙服务请求到达集合计算当前时刻需要的服务功能实例数量; 然后根据现有的服务功能实例数量和所需的服务功能实例数量, 对服务功能实例进行动态配置, 实现虚拟资源的动态调整; 最后根据调整后的 SF 实例数量以及网络状态计算新到达的服务请求的服务功能路径, 按需提供网络服务。

算法 1 在线服务功能链部署算法

输入 $G(V, E), C_v, B_{uv}, L_{uv}, F, ct_f, cr_f, \Gamma, \phi_f, \varphi_f$
输出 服务功能实例部署方案, SFC 请求部署方案

1. 初始化 $t = 0$;
2. for $t \in \Gamma$
3. 读取当前时间间隙 SFC 请求集合;
4. 计算当前时刻服务功能实例需求数量 $z_l^*(t)$;
5. for $i = 1: |F|$
6. 调用在线服务功能实例配置算法对服务功能实例进行自适应扩展;
7. end for
8. 对每个到达的服务请求, 调用基于遗传算法的服务功能路径配置算法计算服务功能路径并进行部署;
9. end for

2.1 在线服务功能实例配置算法

在线服务功能实例配置的目的在于在保证提供网络服务的同时尽可能降低冗余服务功能实例运行开销和服务功能实例的部署开销, 合理配置冗余服务功能实例的生命周期。在流量动态变化的场景下, 无法获得未来时刻的流量变化, 当前时刻过量配置的服务功能实例是否保留需要进行权衡。若将当前时刻冗余的服务功能实例进行立即删除, 则下一时刻流量增加时, 会增加服务功能实例的部署开销; 反之, 若继续保留冗余的服务功能实例, 当下一时刻流量继续减少时, 则持续保留空置状态实例会增加运行开销。为平衡两种开销之间的矛盾, 本文采用雪橇租赁模型^[14]对状态为 idle 的服务功能实例的生命周期进行管理。经典的雪橇租赁问题是指在滑雪

天数未知的情况下, 确定雪橇的租赁和购买策略, 使滑雪支出接近天数已知情况下的最优解。设雪橇购买价格是租赁价格的 m 倍, 则按照式(12)概率分布 p_n 随机选择天数 n , 即租赁 $n-1$ 天, 第 n 天继续滑雪则购买雪橇, 能够达到与滑雪天数已知的情况下相比下的最优竞争比 $e/(e-1)$ 。

$$p_n = \begin{cases} \frac{\left(\frac{m-1}{m}\right)^{m-n}}{m\left(1 - \left(1 - \left(\frac{1}{m}\right)^m\right)\right)}, & n \leq m \\ 0, & n > m \end{cases} \quad (12)$$

在服务功能实例生命周期管理问题中, 未来时刻流量的变化是未知的, 当前时刻 idle 状态的服务功能实例再次被设置为 active 状态所等待的时间也是未知的, 持续 idle 状态会增加运行开销, 若将其删除则会增加潜在的部署开销, 其本质上为雪橇租赁模型。基于该模型的在线服务功能实例配置算法具体如下:

算法 2 在线服务功能实例配置算法

输入 $G(V, E), Z_f(t-1), W_f(t-1), z_f(t-1), w_f(t-1), z_l^*(t)$

输出 当前时刻服务功能实例集合 $Z_f(t), W_f(t)$

1. if $z_l^*(t) \geq z_f(t-1) + w_f(t-1)$ then
2. 部署 $z_l^*(t) - z_f(t-1) - w_f(t-1)$ 个服务功能实例;
3. 将系统中所有类型为 f 的 VNF 实例状态设为 active;
4. 更新底层节点剩余资源 $Z_f(t-1), W_f(t-1)$;
5. else if $z_l^*(t) < z_f(t-1)$ then
6. 对 $Z_f(t-1)$ 中的服务功能实例按照负载大小进行升序排列;
7. 将序列中前 $z_f(t-1) - z_l^*(t)$ 个实例状态设置为 idle, 记录 idle 状态开始时间 $istart$ 并按照式(12)设置生命周期 $deadline$;
8. 更新 $Z_f(t-1), W_f(t-1)$;
9. else
10. 对 $W_f(t-1)$ 中的服务功能实例按照 $istart$ 进行降序排列;
11. 将序列中前 $z_l^*(t) - z_f(t-1)$ 个实例设置为 active 状态;
12. 更新 $Z_f(t-1), W_f(t-1)$;
13. end if
14. end if
15. for 集合 $W_f(t-1)$ 中所有服务功能实例 do
16. if 该实例的 idle 状态时间 $\geq deadline$ then
17. if 该实例上有 SFC 请求的流量承载 then
18. 调用基于遗传算法的服务功能路径配置算法重新计算服务功能路径并进行流量迁移;
19. end if
20. 将该实例删除;
21. end if

22. end for

23. $Z_i(t) = Z_i(t-1)$, $W_i(t) = W_i(t-1)$

在当前时刻所需的实例数量 $z_f^*(t) \geq z_f(t-1) + w_f(t-1)$ 时,则将系统中所有的服务功能实例状态设置为 active,并部署 $z_f^*(t+1) - z_f(t) - w_f(t)$ 个服务功能实例,以满足增长的流量需求(算法 2 中的步骤 1 ~ 步骤 4)。在部署新的服务功能实例时,将该问题建模为背包模型^[17],以尽可能减少部署服务功能实例所占用的物理节点。当 $z_f^*(t) < z_f(t-1)$ 时,对 $Z_f(t-1)$ 中状态为 active 的服务功能实例按照负载进行升序排列,将序列中前 $z_f(t-1) - z_f^*(t)$ 个实例设置为 idle 状态,以在将空闲状态实例删除时尽可能减少需要改变服务功能路径的 SFC 请求数量。对设置为 idle 状态的服务功能实例,将 φ_f/ϕ_f 代入式(12),设置其生命周期 deadline,并记录 idle 状态开始时间 istart(算法 2 中的步骤 5 ~ 步骤 8)。当 $z_f(t-1) \leq z_f^*(t) < z_f(t-1) + w_f(t-1)$ 时,对 $W_f(t-1)$ 中的服务功能实例按照 istart 进行降序排列,将前 $z_f^*(t) - z_f(t-1)$ 个实例由 idle 状态设置为 active 状态,即优先使用 idle 状态持续时间最短的服务功能实例,持续时间长的实例很可能被删除,以减少资源开销(算法 2 中的步骤 9 ~ 步骤 12),每进行一次状态改变均对 $Z_f(t-1)$ 和 $W_f(t-1)$ 进行更新。对 $W_f(t-1)$ 中到达生命周期的服务功能实例,若仍有 SFC 请求部署,则调用基于遗传算法的服务功能路径配置算法重新计算服务功能路径,将数据流量迁移完成后对该实例进行删除(算法 2 中的步骤 16 ~ 步骤 22)。由于 OpenNF^[18] 支持无损和保留数据的流状态迁移,因此算法中不考虑具体的流量状态迁移过程。

2.2 基于遗传算法的服务功能路径配置算法

对于每个到达的 SFC 请求,需要按照请求中的服务功能需求在网络中找到对应的服务功能实例,并且设置服务功能实例之间的路由路径,以实现数据流量的转发,其本质上是 NP 难的组合优化问题,而遗传算法对于求解该类问题非常有效^[19]。本文提出基于遗传算法的 SFC 服务功能路径配置算法,以快速找到服务请求的最短服务功能路径。

1) 染色体编码:每个染色体采用一个行向量进行表示,向量的列数为该 SFC 中包含的服务功能数量,向量中第 j 个元素的数值代表了 g_{ij} 所对应的服务功能实例序号。向量中每个元素取值范围为 $Z_f(t)$ 中剩余吞吐量大于 s_i 请求带宽的实例构成的集合 $avail_f(t)$ 。

2) 染色体解码:对应编码方案,首先确定染色体向量中每个元素对应的服务功能实例所在的节点位置;然后源节点、服务功能实例承载节点、目的节点

之间流量路由采用最短路径算法,计算得到该染色体对应的 SFC 部署和路由方案。若相邻的服务功能实例部署在同一物理节点,则该段路径长度为 0。

3) 适应度计算:适应度的计算方法如式(13)所示。

$$f(pop_i) = \frac{1}{Lr_i + \varepsilon} \quad (13)$$

其中, Lr_i 表示方案 pop_i 对应的路径长度, ε 为常数。适应度值越大,说明该方案路径长度越小。

4) 选择复制:采用轮盘赌方法进行选择,每个染色体被选中的概率为 $p(pop_i) = f(pop_i) / \sum_i f(pop_i)$ 。

此外,采用单点交叉法和单点变异法进行交叉和变异操作。算法过程具体如下:

算法 3 基于 GA 的服务功能路径配置算法

输入 $s_i, G(V, E), Z_f(t)$ 、种群大小 $Popsiz$ 、交叉概率 P_c 、变异概率 P_m 、连续最优次数 Opt

输出 SFC 部署方案 pop_{best} 、服务功能实例负载状态、链路剩余资源

1. 根据 s_i 请求带宽和 $Z_f(t)$ 中实例剩余吞吐量计算可承载该请求的实例集合 $avail_f(t)$;
2. 根据 $avail_f(t), G(V, E)$ 生成初始种群 Pop , 设置 $n = 0$, 最优适应度值 $f_{best} = 0, f'_{best} = 0$;
3. while $n \leq Opt$ do
4. 对每个 $pop_i \in Pop$, 计算适应度 $f(pop_i)$;
5. 得到 pop_{best} 和 f_{best} ;
6. 采用轮盘赌方法进行复制构成新种群 $Newpop$;
7. 对种群中的染色体以概率 P_c 进行交叉;
8. 对种群中的染色体以概率 P_m 进行变异;
9. if $f_{best} = f'_{best}$ then
10. $n = n + 1$;
11. else
12. $n = 0$;
13. end if
14. $f'_{best} = f_{best}$;
15. $Pop = Newpop$
16. end while
17. 根据 pop_{best} 对 s_i 进行部署, 更新 $Z_f(t)$ 中实例剩余吞吐量和链路剩余带宽

3 实验结果与分析

3.1 实验设置

实验在 Intel Corei7-7500U、16 GB RAM 笔记本电脑上进行,程序基于 Matlab 2018b 实现。由于在数据中心网络中大多为胖树结构,各机架之间可以相互连接,底层网络视图为 10 个节点的全连接视图,每个节点可虚拟出的 CPU 数量设置为 48,每条链路带宽设置为 4 Gb/s。为更加真实地验证算法,服务请求强度从试验网抓包分析获得,每隔 15 min

对抓取的数据包进行分析,通过对比与前一时间段不同起始 IP 数量和协议类型获取该时间段内请求到达强度,并将每 15 min 设定为一个单位时间,生成服务请求到达强度的时间序列,该数据在 GitHub^[20] 上进行公开。每个 SFC 请求带宽在 [30 Mb/s, 50 Mb/s] 之间均匀分布,持续时间服从 $\lambda = 1$ 的负指数分布,生成的网络流量如图 2 所示。

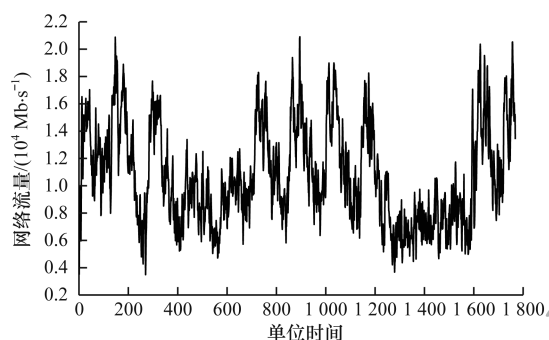


图2 网络流量变化情况

所有 SFC 请求的服务功能序列统一设置为 Firewall→IDS→Proxy, 每种服务功能实例占用的 CPU 资源和吞吐量^[21] 如表 1 所示。

表 1 各服务功能 CPU 需求数量和吞吐量

服务功能类型	CPU 需求数量	吞吐量/(Mb · s ⁻¹)
Firewall	4	900
Proxy	4	900
IDS	8	600

为在仿真过程中获得每个时刻服务功能实例需求数量,采用式(14)进行换算, $\lceil \cdot \rceil$ 符号为向上取整。

$$CR_f(t) = \left\lceil \sum_{s_i \in S(t)} b_i / ct_f \right\rceil \quad (14)$$

3.2 性能分析

对不同的服务功能实例生命周期配置策略的开销进行对比分析。设置部署开销系数 $\phi_d = 7$, 运行开销系数 $\phi_r = 1$, 本文提出的基于雪橇租赁模型的 idle 状态实例生命周期管理方法 (ski07) 与在 [1, 7] 区间中随机选择生命周期方法 (rand07)、即刻删除方法 (static01) 和文献 [13] 中的固定设置生命周期方法 (实验中设置为 7, static07) 进行对比。

图 3 为本文提出方法中 Firewall 实例的变化曲线。可以看出, 防火墙总量变化与流量变化趋势整体相同, 但其变化剧烈程度较低, 短期内呈现水平阶梯状, 其原因为网络流量在较短时间内呈现出无规律性, 而 idle 状态实例的存在, 使得已经创建的服务功能实例能够重新设置为 active 状态进行使用, 避免服务功能实例大量创建, 造成实例数量的剧烈变化。

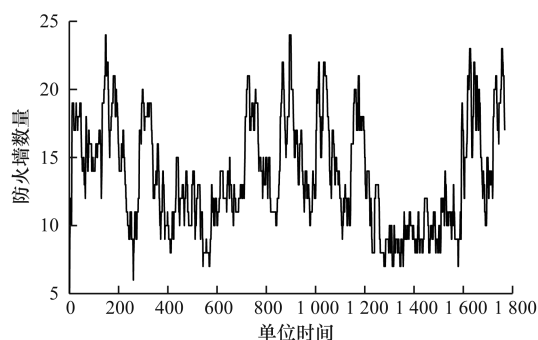


图3 防火墙数量变化情况

图 4 为 idle 状态实例运行开销曲线, static01 方法对过量配置的服务功能立即进行删除, 其运行开销始终为 0。然而, 由于网络流量的剧烈变化, 当未来时刻需要更多的服务功能实例满足服务请求需求时, 则会部署未来时刻需要的更多服务功能实例, 因此图 5 中相比于其他保留空闲状态实例的方法, 其对应的部署开销最高。static07 方法始终设定 idle 状态的服务功能实例生命周期为固定值, 则会造成系统中驻留大量的冗余实例, 占用大量的虚拟资源, 造成运行开销最高, 但是预留冗余实例减少了未来时刻流量变化的过程中新服务功能实例的创建, 因此其对应的部署开销最低。基于随机选择的生命周期配置方法 rand07 在 [1, 7] 区间中随机配置服务功能实例的生命周期, 基于雪橇租赁模型的 ski07 方法采用式 (12) 对 idle 状态的实例进行生命周期配置, 随着自变量 n 在 [1, m] 区间中增加, 概率 p_n 也相应增加, 即 ski07 方法与 rand07 方法相比, 对于每一个 idle 状态的服务功能实例, 其更倾向于设置更长的生命周期, 因此在图 4 中, 其运行开销比 rand07 方法要大。较长的生命周期意味着空闲状态服务功能实例在系统中驻留时间更长, 在未来时刻需要更多的服务功能实例时, 其能够转换为 active 状态, 为数据流量提供服务, 而不需要创建更多的新实例, 因此在图 5 中 ski07 方法比 rand07 方法部署开销小。

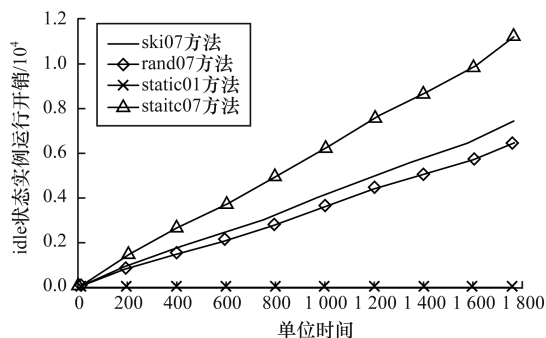


图4 idle 状态实例运行开销

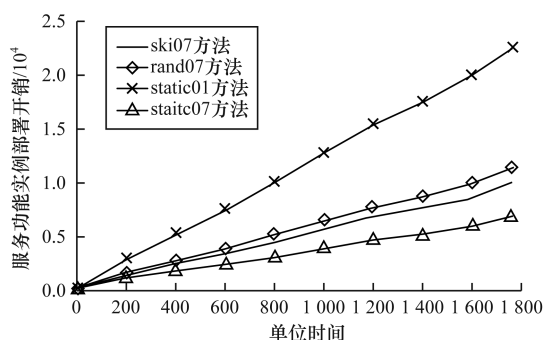


图5 服务功能实例部署开销

图6为总的自适应扩展开销变化。可以看出,ski07方法所产生的总开销最小,比rand07方法降低了1.8%~2.0%,并且随着时间的增加,各方法之间产生的开销差异也在逐渐加大。

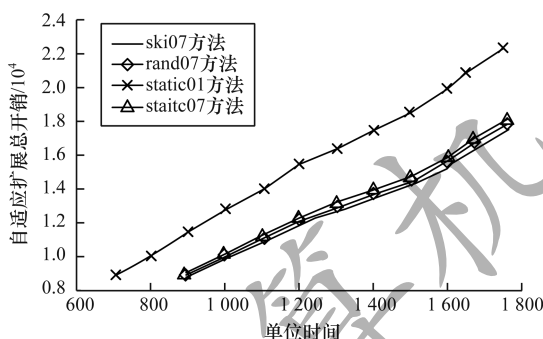


图6 自适应扩展总开销

将本文提出的基于GA算法的最短服务功能路径配置方法与OpenDaylight^[22]中的随机选择算法(RAND)和负载均衡算法(LB)的带宽利用率进行对比,如图7所示。可以看出,本文方法能显著降低带宽的使用,尤其在大流量情况下,比其他方法降低了4%~7%。主要原因为在大流量条件下,虚拟机数量增加且集中部署在部分节点上,算法更倾向于将服务功能链部署在位于同一底层节点或者少量节点的多个虚拟机上,不同服务功能实例之间的流量传输将在节点内部虚拟交换接口中进行,从而大幅减少节点之间带宽资源的使用,而其他两种方法不能很好地利用服务功能实例集中的部署优势,其带宽利用率较高。

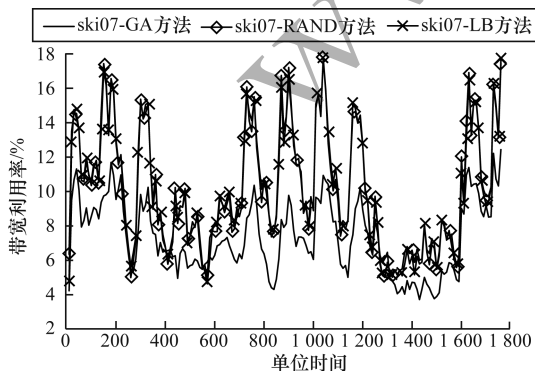


图7 带宽利用率变化情况

4 结束语

本文研究服务功能链请求到达场景下服务功能实例的动态扩展和服务功能链的动态部署问题,提出基于雪橇租赁模型的服务功能实例生命周期管理和在线部署方法,以及基于遗传算法的服务功能路径配置算法,降低了动态场景下服务功能自适应扩展开销和服务功能链的部署带宽,为运营商实现云网一体化的自动化运营提供了参考。下一步将结合5G发展需求,研究5G核心网中服务功能链的动态部署方案。

参考文献

- [1] European Telecommunications Standards Institute. Network function virtualization-white paper #3 [EB/OL]. [2019-07-08]. http://portal.etsi.org/NFV/NFV_White_Paper3.pdf.
- [2] GUICHARD J, QUINN P. Service Function Chaining (SFC) architecture: 7665 [EB/OL]. [2019-07-08]. <https://datatracker.ietf.org/doc/rfc7665/>.
- [3] SHERRY J, HASAN S, SCOTT C, et al. Making middleboxes someone else's problem: network processing as a cloud service [J]. ACM SIGCOMM Computer Communication Review, 2012, 42(4): 13-24.
- [4] SHERRY J, RATNASAMY S, AT J S. A survey of enterprise middlebox deployments: UCB/EECS-2012-24 [R]. Oakland, USA: University of California, 2012.
- [5] KREUTZ D, RAMOS F M V, VERISSIMO P, et al. Software-defined networking: a comprehensive survey [J]. Proceedings of the IEEE, 2015, 103(1): 14-76.
- [6] DE SOUSA N F S, PEREZ D A L, ROSA R V, et al. Network service orchestration: a survey [J]. Computer Communications, 2019, 142: 69-94.
- [7] HANTOUTI H, BENAMAR N, TALEB T, et al. Traffic steering for service function chaining [J]. IEEE Communications Surveys and Tutorials, 2018, 21(1): 487-507.
- [8] SHI Jiugen, ZHANG Jing, XU Hao, et al. Joint optimization of virtualized network function placement and routing allocation for operational expenditure [J]. Journal of Electronics and Information Technology, 2019, 41(4): 973-979. (in Chinese)
史久根, 张径, 徐皓, 等. 一种面向运营成本优化的虚拟网络功能部署和路由分配策略 [J]. 电子与信息学报, 2019, 41(4): 973-979.
- [9] YUAN Quan, TANG Hongbo, HUANG Kaizhi, et al. Deployment method for vEPC virtualized network function via Q-learning [J]. Journal on Communications, 2017, 38(8): 172-182. (in Chinese)
袁泉, 汤红波, 黄开枝, 等. 基于 Q-learning 算法的 vEPC 虚拟网络功能部署方法 [J]. 通信学报, 2017, 38(8): 172-182.
- [10] WANG Xiaoke, WU Chuan, LE F, et al. Online VNF scaling in datacenters [C] // Proceedings of the 9th International Conference on Cloud Computing. San Francisco, USA: IEEE Press, 2016: 140-147.

- [11] TANG Lun, ZHOU Yu, YANG Youchao, et al. Virtual network function dynamic deployment algorithm based on prediction for 5G network slicing [J]. Journal of Electronics and Information Technology, 2019, 41(9): 2071-2078. (in Chinese)
唐伦, 周钰, 杨友超, 等. 5G 网络切片场景中基于预测的虚拟网络功能动态部署算法[J]. 电子与信息学报, 2019, 41(9): 2071-2078.
- [12] WANG Huijian, LIU Zheng, LI Yun, et al. Trend prediction method of time series trends based on neural network language model [J]. Computer Engineering, 2019, 45(7): 13-19, 25. (in Chinese)
王慧健, 刘峥, 李云, 等. 基于神经网络语言模型的时间序列趋势预测方法[J]. 计算机工程, 2019, 45(7): 13-19, 25.
- [13] FEI Xincan, LIU Fangming, XU Hong, et al. Adaptive VNF scaling and flow routing with proactive demand prediction [C]//Proceedings of IEEE Conference on Computer Communications. Washington D. C., USA: IEEE Press, 2018: 486-494.
- [14] KARLIN A R, MANASSE M S, MCGEOCH L A, et al. Competitive randomized algorithms for nonuniform problems [J]. Algorithmica, 1994, 11(6): 542-571.
- [15] YI Bo, WANG Xingwei, LI Keqin, et al. A comprehensive survey of network function virtualization [J]. Computer Networks, 2018, 133: 212-262.
- [16] TOOSI A N, SON Junming, CHI Qinghua, et al. ElasticSFC: auto-scaling techniques for elastic service function chaining in network functions virtualization-based clouds [J]. Journal of Systems and Software, 2019, 152: 108-119.
- [17] FRIESEN D K, LANGSTON M A. Variable sized bin packing [J]. SIAM Journal on Computing, 1986, 15(1): 222-230.
- [18] GEMBER-JACOBSON A, VISWANATHAN R, PRAKASH C, et al. OpenNF: enabling innovation in network function control [J]. ACM SIGCOMM Computer Communication Review, 2014, 44(4): 163-174.
- [19] CAO Kai, CHEN Guohu, JIANG Hua, et al. Guided self-adaptive evolutionary genetic algorithm [J]. Journal of Electronics and Information Technology, 2014, 36(8): 1884-1890. (in Chinese)
曹凯, 陈国虎, 江桦, 等. 自适应引导进化遗传算法[J]. 电子与信息学报, 2014, 36(8): 1884-1890.
- [20] Network-traffic-experiment [EB/OL]. [2019-07-08]. <https://github.com/lovesmile616/network-traffic-experiment.git>.
- [21] BARI M F, CHOWDHURY S R, AHMED R, et al. On orchestrating virtual network functions in NFV [C]//Proceedings of the 11th International Conference on Network and Service Management. Washington D. C., USA: IEEE Press, 2015: 50-56.
- [22] OpenDaylight user guide [EB/OL]. [2019-07-08]. <https://nexus.opendaylight.org/content/sites/site/org.opendaylight/docs/master/userguide/manuals/userguide/bk-user-guide/content/>.

编辑 陆燕菲