基于成本及功耗联合优化的SDN虚拟网络映射算法

柴 蓉,谢德胜,陈前斌
(重庆邮电大学通信与信息工程学院,重庆 400065)

摘 要: 针对多个虚拟网络请求(Virtual Network Request, VNR)动态到达的网络场景,本文提出一种基于成本及功耗联合优化的软件定义网络(Software-Defined Networking, SDN)虚拟网络映射(Virtual Network Embedding, VNE)算法.在对虚拟节点及链路映射成本及功耗进行评估的基础上,建模VNE成本及功耗的代价函数,进而在满足资源需求等约束条件下,建模基于代价函数最小化的VNE模型.该优化问题为整数线性规划问题,难以直接求解;为解决此问题,提出基于时间窗的虚拟网络批处理映射策略动态处理在线请求.继而针对特定时间窗内的VNR,将其转换为虚拟节点映射子问题和虚拟链路映射子问题,并应用启发式算法对两个子问题分别进行求解,从而确定VNR映射策略.仿真结果表明,所提算法能显著减少VNE成本及功耗,提高VNR接受率.

关键词: 软件定义网络; 虚拟网络映射; 联合优化; 启发式算法
中图分类号: TP915 **文献标识码:** A **文章编号:** 0372-2112(2021)08-1615-10
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.12263/DZXB.20190688

Cost and Power Consumption Joint Optimization Based Virtual Network Embedding Algorithm for Software-Defined Networking

CHAI Rong, XIE De-sheng, CHEN Qian-bin
(*School of Communication and Information Engineering, Chongqing University of Post and Telecommunications, Chongqing* 400065, *China*)

Abstract: For the network scenario where multiple virtual network requests (VNRs) arrive dynamically, a cost and power consumption joint optimization based virtual network embedding (VNE) algorithm was proposed for software-defined networking (SDN). Based on the evaluation of the cost and power consumption required for embedding virtual nodes and links, the cost function of VNE was formulated. Under the constraints of resource requirements, the VNE problem was formulated as cost function minimization problem. A time window-based batch embedding strategy was proposed to dynamically process online VNRs. The VNR in certain time window was transformed into virtual node embedding subproblem and virtual link embedding subproblem, and the corresponding heuristic algorithms were proposed, respectively. Simulation results showed that the proposed algorithm reduced the cost and power consumption of VNRs, and improved the acceptance ratio of VNRs.

Key words: software-defined networking; virtual network embedding; joint optimization; heuristic algorithm

1 引言

云计算、大数据及物联网等应用的多样化业务需求对传统网络架构及管理机制提出挑战,针对这一问题,软件定义网络(Software-Defined Networking, SDN)应运而生[1].作为未来网络体系架构的重要技术之一,SDN将网络的控制层与数据层相分离,实现了对网络的可编程集中控制及高效管理.在支持网络虚拟化的SDN中[2],如何实现高效虚拟网络映射(Virtual Network Embedding, VNE),即如何将用户的虚拟网络请求(Virtual Network Request, VNR)映射至底层网络是亟待解决的重要问题[3,4].

目前已有研究考虑VNE算法的优化设计,旨在降低VNE开销或提高请求接受率,实现底层资源的高效利用[5~8].为实现底层网络负载,提升VNE性能,文献[9~11]提出基于负载优化的VNE策略.文献[12~14]研究能量有效的VNE问题,提出基于映射能耗优化的

收稿日期:2019-06-18;修回日期:2020-11-01;责任编辑:孙瑶
基金项目:国家自然科学基金(No. 62071078)

VNE策略. 文献[15,16]联合考虑VNE及虚拟网络迁移,以实现网络资源高效利用及长期VNE性能优化. 针对VNR动态到达的网络场景,文献[17~22]研究在线VNE问题. 文献[17,18]采用先到先映射机制,根据到达时间对VNR进行实时映射;文献[19,20]采用批处理映射机制,收集一段时间内到达的VNR,根据优先级对VNR进行排序及映射;文献[21,22]基于强化学习方法提升多层、多阶段VNE算法性能.

现有VNE算法研究较为孤立地考虑VNE成本及功耗等因素,未充分考虑VNR动态特征、映射成本及功耗等多个因素对映射策略的影响,导致VNE策略的性能受限. 本文针对VNR动态到达的SDN场景,联合考虑VNE成本及功耗,提出一种基于联合优化的VNE算法.

2 网络模型

2.1 底层网络

本文考虑多个VNR动态到达底层网络,其中,底层网络由M个底层节点及连接节点的底层链路组成. 方便起见,本文将底层网络拓扑抽象为带权无向图$G^{\mathrm{s}}=(N^{\mathrm{s}},E^{\mathrm{s}},A_N^{\mathrm{s}},A_E^{\mathrm{s}})$,其中,$N^{\mathrm{s}}=\{n_i^{\mathrm{s}},1\leqslant i\leqslant M\}$表示底层节点集合,$E^{\mathrm{s}}=\{e_{i,j}^{\mathrm{s}},1\leqslant i,j\leqslant M,i\neq j\}$表示底层链路集合,$A_N^{\mathrm{s}}=\{(C(n_i^{\mathrm{s}}),T(n_i^{\mathrm{s}})),n_i^{\mathrm{s}}\in N^{\mathrm{s}}\}$表示底层节点属性集合,$n_i^{\mathrm{s}}$,$e_{i,j}^{\mathrm{s}}$,$C(n_i^{\mathrm{s}})$,$T(n_i^{\mathrm{s}})$,$B(e_{i,j}^{\mathrm{s}})$及$D(e_{i,j}^{\mathrm{s}})$的含义如表1所示.

本文研究特定时段内的VNE问题,将给定时段划分为多个等长时隙. 为降低底层节点及链路功耗,假设底层网络支持休眠机制,即任一底层节点及链路均可处于激活或休眠两种状态. 若底层节点(链路)未承载任何虚拟节点(链路),则该底层节点(链路)处于休眠状态;否则,处于激活状态. 为表征底层节点(链路)在某时隙的状态,引入底层节点(链路)状态标识符,令$x_{i,t}$($x_{i,j,t}$)表示t时隙n_i^{s}($e_{i,j}^{\mathrm{s}}$)的状态标识符,若t时隙n_i^{s}($e_{i,j}^{\mathrm{s}}$)处于激活状态,则$x_{i,t}=1$($x_{i,j,t}=1$),否则,$x_{i,t}=0$($x_{i,j,t}=0$).

2.2 虚拟网络请求

本文假设在给定时段,可能存在多个VNR动态到达;同时,已到达的VNR可能在某时隙失效. 假定VNR到达服从泊松过程. 若存在新到达VNR,则在该请求失效时间之前均可执行映射,为其分配相应的底层资源. 对于已成功映射的VNR,若其失效时间到达,则需释放其占用的底层资源.

令K表示VNR数量,并将第k个VNR标记为$\mathrm{VNR}_k=(G_k^{\mathrm{v}},t_k^a,t_k^d)$,其中,$G_k^{\mathrm{v}}$表示$\mathrm{VNR}_k$的虚拟网络拓扑,采用带权无向图$G_k^{\mathrm{v}}=(N_k^{\mathrm{v}},E_k^{\mathrm{v}},A_{k,N}^{\mathrm{v}},A_{k,E}^{\mathrm{v}})$表示,其中,$N_k^{\mathrm{v}}=\{n_{k,u}^{\mathrm{v}}\}$及$E_k^{\mathrm{v}}=\{e_{k,u,r}^{\mathrm{v}}\}$分别表示$G_k^{\mathrm{v}}$中虚拟节点及虚拟链路集合,$A_{k,N}^{\mathrm{v}}=\{(C(n_{k,u}^{\mathrm{v}}),T(n_{k,u}^{\mathrm{v}}))\}$及$A_{k,E}^{\mathrm{v}}=\{(B(e_{k,u,r}^{\mathrm{v}}),D(e_{k,u,r}^{\mathrm{v}}))\}$分别表示虚拟节点及虚拟链路属性集合,其中,$M_k$表示$G_k^{\mathrm{v}}$中虚拟节点数量,$n_{k,u}^{\mathrm{v}}$,$e_{k,u,r}^{\mathrm{v}}$,$C(n_{k,u}^{\mathrm{v}})$,$T(n_{k,u}^{\mathrm{v}})$,$B(e_{k,u,r}^{\mathrm{v}})$及$D(e_{k,u,r}^{\mathrm{v}})$的含义如表1所示. 令$t_k^a(t_k^d)$表示$\mathrm{VNR}_k$的到达(失效)时间,定义$y_{k,t}^a(y_{k,t}^d)$为$\mathrm{VNR}_k$的到达(失效)标识符,若$t=t_k^a$,$y_{k,t}^a=1$,否则,$y_{k,t}^a=0$;若$t\geqslant t_k^d$,$y_{k,t}^d=1$,否则,$y_{k,t}^d=0$.

表1 文中主要符号及其含义

符号	含义	符号	含义	符号	含义
n_i^{s}	第i个底层节点	$B(e_{k,u,r}^{\mathrm{v}})$	$e_{k,u,r}^{\mathrm{v}}$带宽容量需求	$\alpha_{k,u,i}$	节点映射标识符
$e_{i,j}^{\mathrm{s}}$	n_i^{s}与n_j^{s}之间链路	$D(e_{k,u,r}^{\mathrm{v}})$	$e_{k,u,r}^{\mathrm{v}}$的最大可容忍链路传播时延	$\beta_{k,u,r,i,j}$	链路映射标识符
$C(n_i^{\mathrm{s}})$	n_i^{s}的CPU初始容量	$C(n_{k,u}^{\mathrm{v}})$	$n_{k,u}^{\mathrm{v}}$的CPU容量需求	t_k^a	VNR_k到达时间
$C_t(n_i^{\mathrm{s}})$	t时隙n_i^{s}已使用的CPU容量	Θ	VNE代价函数	t_k^d	VNR_k失效时间
$\hat{C}_t(n_i^{\mathrm{s}})$	t时隙n_i^{s}的可用CPU容量	$\Theta_{k,t}$	t时隙VNR_k映射代价	$y_{k,t}^a$	VNR_k到达标识符
$T(n_i^{\mathrm{s}})$	n_i^{s}的TCAM初始容量	$\Theta_{k,t}^n$	VNR_k虚拟节点映射代价	$y_{k,t}^d$	VNR_k失效标识符
$T_t(n_i^{\mathrm{s}})$	t时隙n_i^{s}已使用的TCAM容量	$Q_{k,t}$	t时隙VNR_k映射成本	$\psi_t^{\mathrm{s}}(n_i^{\mathrm{s}})$	n_i^{s}度量值
$\hat{T}_t(n_i^{\mathrm{s}})$	t时隙n_i^{s}的可用TCAM容量	$Q_{k,t}^n$	t时隙VNR_k节点映射成本	$\psi^{\mathrm{v}}(n_{k,u}^{\mathrm{v}})$	$n_{k,u}^{\mathrm{v}}$度量值
$B(e_{i,j}^{\mathrm{s}})$	$e_{i,j}^{\mathrm{s}}$的带宽初始容量	$Q_{k,t}^e$	t时隙VNR_k链路映射成本	$\varphi_t(n_i^{\mathrm{s}})$	t时隙n_i^{s}映射能力
$B_t(e_{i,j}^{\mathrm{s}})$	t时隙$e_{i,j}^{\mathrm{s}}$已使用带宽容量	$P_{k,t}$	t时隙VNR_k映射功耗	$\varphi_t(e_{i,j}^{\mathrm{s}})$	t时隙$e_{i,j}^{\mathrm{s}}$映射能力
$\hat{B}_t(e_{i,j}^{\mathrm{s}})$	t时隙$e_{i,j}^{\mathrm{s}}$可用带宽容量	$P_{k,t}^n$	t时隙VNR_k映射底层节点功耗	$q_t(n_i^{\mathrm{s}})$	t时隙n_i^{s}资源度量值
$D(e_{i,j}^{\mathrm{s}})$	$e_{i,j}^{\mathrm{s}}$链路传播时延	$P_{k,t}^e$	t时隙VNR_k映射底层链路功耗	$p_t(n_i^{\mathrm{s}})$	t时隙n_i^{s}功耗度量值
$n_{k,u}^{\mathrm{v}}$	G_k^{v}第u个虚拟节点	$\Gamma_t^{\mathrm{C}}(n_i^{\mathrm{s}})$	t时隙n_i^{s}单位CPU资源成本函数	$\varpi_t^{(k,u,r,i,j)}$	t时隙$e_{k,u,r}^{\mathrm{v}}$映射至$e_{i,j}^{\mathrm{s}}$的代价
$e_{k,u,r}^{\mathrm{v}}$	$n_{k,u}^{\mathrm{v}}$与$n_{k,r}^{\mathrm{v}}$之间虚拟链路	$\Gamma_t^{\mathrm{T}}(n_i^{\mathrm{s}})$	t时隙n_i^{s}单位TCAM资源成本函数	$\varpi_t^{(k,u,r,i)}$	t时隙$e_{k,u,r}^{\mathrm{v}}$映射至底层路径中间节点激活功耗
$T(n_{k,u}^{\mathrm{v}})$	$n_{k,u}^{\mathrm{v}}$的TCAM容量需求	$\Gamma_t^{\mathrm{B}}(e_{i,j}^{\mathrm{s}})$	t时隙$e_{i,j}^{\mathrm{s}}$单位带宽资源成本函数	$R(\mathrm{VNR}_k)$	VNR_k所需底层资源总量
$x_{i,t}$	t时隙n_i^{s}状态标识符	$\eta_{i,t}^{\mathrm{C}}$	t时隙n_i^{s}的CPU负载率	$\Delta\eta_i$	$n_{k,u}^{\mathrm{v}}$映射至n_i^{s}节点负载率
$x_{i,j,t}$	t时隙$e_{i,j}^{\mathrm{s}}$状态标识符	$\eta_{i,t}^{\mathrm{T}}$	t时隙n_i^{s}的TCAM负载率	$\eta_{i,j,t}^{\mathrm{B}}$	t时隙$e_{i,j}^{\mathrm{s}}$带宽负载率

图1所示为本文考虑的系统模型图,图中以两个VNR到达为例说明VNR映射过程. 假设当$t = t_1^a$时,VNR_1到达,需为VNR_1分配底层节点及链路执行VNE,此时承载VNR_1的底层节点及链路从休眠状态切换至激活状态;当$t = t_1^d$时,VNR_1失效,释放VNR_1占用的底层资源. 类似地,需实现VNR_2的映射及资源释放.

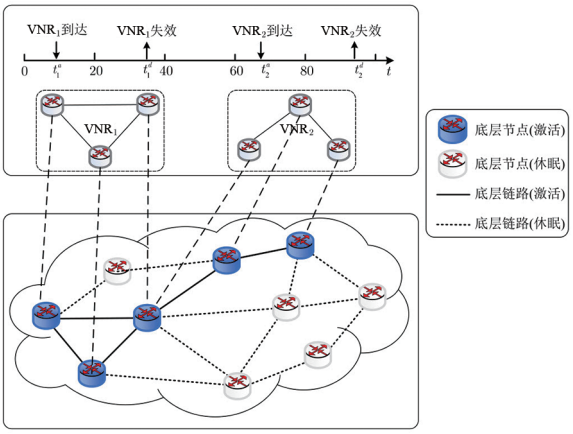


图1　系统模型图

3　优化问题建模

本文综合考虑VNR的映射成本及底层网络功耗,定义VNE代价函数为映射成本及底层网络功耗的线性加权函数,进而以代价函数最小化为目标建模VNE问题.

3.1　代价函数建模

本文建模时段T内VNE代价函数为各时隙所有VNR映射代价的总和,即

$$\Theta = \sum_{k=1}^{K}\sum_{t=1}^{T}\Theta_{k,t} \tag{1}$$

其中,$\Theta_{k,t}$表示t时隙VNR_k的映射代价,建模为

$$\Theta_{k,t} = \mu_Q Q_{k,t} + \mu_P P_{k,t} \tag{2}$$

式中,$Q_{k,t}$及$P_{k,t}$分别表示t时隙VNR_k的映射成本及对应底层网络功耗,μ_Q和μ_P分别表示映射成本及底层网络功耗的权值因子,$\mu_Q,\mu_P \geqslant 0$. 需说明的是,为综合考虑VNR_k的映射成本及功耗,实现映射代价优化,应合理选择μ_Q和μ_P,使得$\mu_Q Q_{k,t}$与$\mu_P P_{k,t}$具有可比性. 式(2)中,$Q_{k,t}$由节点及链路映射成本组成,即

$$Q_{k,t} = Q_{k,t}^n + Q_{k,t}^e \tag{3}$$

其中,$Q_{k,t}^n$为t时隙VNR_k的节点映射成本,建模为

$$Q_{k,t}^n = \sum_{n_{k,u}^v \in N_k^v}\sum_{n_i^s \in N^s}\alpha_{k,u,i}(\Gamma_t^C(n_i^s)C(n_{k,u}^v) + \Gamma_t^T(n_i^s)T(n_{k,u}^v)) \tag{4}$$

式中,$\alpha_{k,u,i} \in \{0,1\}$表示节点映射标识符,若$n_{k,u}^v$映射至$n_i^s$,则$\alpha_{k,u,i} = 1$;反之,$\alpha_{k,u,i} = 0$,$\Gamma_t^C(n_i^s)$和$\Gamma_t^T(n_i^s)$分别表示$t$时隙$n_i^s$的单位中心处理单元(Central Processing Unit,CPU)资源成本函数及单位和三态内容寻址存储器(Ternary Content Addressable Memory, TCAM)资源成本函数. 本文引入Sigmoid函数[23,24]建模$\Gamma_t^C(n_i^s)$及$\Gamma_t^T(n_i^s)$,$\Gamma_t^C(n_i^s)$表示为

$$\Gamma_t^C(n_i^s) = \frac{\lambda^C}{1 + \exp(-\delta^C \eta_{i,t}^C)} \tag{5}$$

其中,λ^C为CPU资源成本系数,δ_C为常量,$\eta_{i,t}^C$表示t时隙n_i^s的CPU负载率,建模为$\eta_{i,t}^C = \dfrac{C_t(n_i^s)}{C(n_i^s)}$,其中,$C_t(n_i^s)$表示$t$时隙$n_i^s$已使用的CPU容量,建模为

$$C_t(n_i^s) = \sum_{k=1}^{K}\sum_{t_0=1}^{t} y_{k,t_0}^a\left(1 - \sum_{t_1=1}^{t} y_{k,t_1}^d\right)\left(\sum_{n_{k,u}^v \in N_k^v}\alpha_{k,u,i}C(n_{k,u}^v)\right) \tag{6}$$

需指出的是,Sigmoid函数广泛用于建模效用函数或成本函数[23]. 根据$\eta_{i,t}^C$的定义可知,较小的$\eta_{i,t}^C$表示n_i^s已使用CPU资源较少,即资源相对充裕,负载较轻;若$\eta_{i,t}^C$取较大值,表示n_i^s已使用较多CPU资源,即负载过重,实现虚拟节点映射难度提升. 根据式(5),$\eta_{i,t}^C$越大,即节点可用资源越少,$\Gamma_t^C(n_i^s)$越大,即使用n_i^s所需支付资源成本更高. 采用类似方法,可建模$\Gamma_t^T(n_i^s)$.

式(3)中,$Q_{k,t}^e$表示t时隙VNR_k的链路映射成本,可建模$Q_{k,t}^e$为

$$Q_{k,t}^e = \sum_{e_{k,u,r}^v \in E_k^v}\sum_{e_{i,j}^s \in E^s}\beta_{k,u,r,i,j}\Gamma_t^B(e_{i,j}^s)B(e_{k,u,r}^v) \tag{7}$$

式中,$\beta_{k,u,r,i,j} \in \{0,1\}$表示链路映射标识符,$\Gamma_t^B(e_{i,j}^s)$表示$t$时隙$e_{i,j}^s$的单位带宽资源成本函数,建模为

$$\Gamma_t^B(e_{i,j}^s) = \frac{\lambda^B}{1 + \exp(-\delta^B \eta_{i,j,t}^B)} \tag{8}$$

其中,λ^B为带宽资源成本系数,δ^B为常量,$\eta_{i,j,t}^B$表示t时隙$e_{i,j}^s$的带宽负载率,建模为$\eta_{i,j,t}^B = \dfrac{B_t(e_{i,j}^s)}{B(e_{i,j}^s)}$,其中,$B_t(e_{i,j}^s)$表示$t$时隙$e_{i,j}^s$已使用的带宽容量,建模为

$$B_t(e_{i,j}^s) = \sum_{k=1}^{K}\sum_{t_0=1}^{t} y_{k,t_0}^a\left(1 - \sum_{t_1=1}^{t} y_{k,t_1}^d\right)\left(\sum_{e_{k,u,r}^v \in E_k^v}\beta_{k,u,r,i,j}B(e_{k,u,r}^v)\right) \tag{9}$$

式(2)中,t时隙实现VNR_k映射底层网络功耗$P_{k,t}$由底层节点及链路功耗组成,即

$$P_{k,t} = P_{k,t}^n + P_{k,t}^e \tag{10}$$

其中,$P_{k,t}^n$表示t时隙VNR_k映射对应底层节点功耗,建模如式(11)所示.

$$P_{k,t}^n = \sum_{n_{k,u}^v \in N_k^v}\sum_{n_i^s \in N^s}\left[\alpha_{k,u,i}((1 - x_{i,t})P_i^{on} + (P_i^{max} - P_i^{on})(2\Delta\eta_i - (\Delta\eta_i)^r))\right] \tag{11}$$

式(11)中,P_i^{on}表示n_i^{s}由休眠状态切换至激活状态所需基础功耗,$P_i^{\max}$表示n_i^{s}满载工作时对应的功耗,$\Delta\eta_i$表示$n_{k,u}^{\text{v}}$映射至n_i^{s}时对应的节点负载率,r为最小化平方误差的校准参数.$\Delta\eta_i$可建模为

$$\Delta\eta_i=\delta_{\text{C}}\frac{C(n_{k,u}^{\text{v}})}{C(n_i^{\text{s}})}+\delta_{\text{T}}\frac{T(n_{k,u}^{\text{v}})}{T(n_i^{\text{s}})} \tag{12}$$

其中,δ_{C}及δ_{T}为权值因子,$0<\delta_{\text{C}},\delta_{\text{T}}<1$,满足条件$\delta_{\text{C}}+\delta_{\text{T}}=1$.

式(10)中,$P_{k,t}^{\text{e}}$表示t时隙VNR_k映射对应的底层链路功耗,建模如式(13)所示.

$$P_{k,t}^{\text{e}}=\sum_{e_{k,u,r}^{\text{v}}\in E_k^{\text{v}}}\sum_{e_{i,j}^{\text{s}}\in E^{\text{s}}}\left[\beta_{k,u,r,i,j}((1-x_{i,j,t})P_{i,j}^{\text{on}}+(P_{i,j}^{\max}-P_{i,j}^{\text{on}})\Delta\eta_{i,j})\right]+\sum_{e_{k,u,r}^{\text{v}}\in E_k^{\text{v}}}\sum_{n_i^{\text{s}}\in N^{\text{s}}}\gamma_{k,u,r,i}(1-x_{i,t})P_i^{\text{on}} \tag{13}$$

式(13)中,$P_{i,j}^{\text{on}}$表示$e_{i,j}^{\text{s}}$由休眠状态切换至激活状态所需基础功耗,$P_{i,j}^{\max}$表示$e_{i,j}^{\text{s}}$满载工作时对应功耗,$\Delta\eta_{i,j}$表示$e_{k,u,r}^{\text{v}}$映射至$e_{i,j}^{\text{s}}$时对应的链路负载率,$\gamma_{k,u,r,i}\in\{0,1\}$表示虚拟链路映射至底层路径时,底层路径的中间节点映射标识符,若n_i^{s}为承载$e_{k,u,r}^{\text{v}}$的底层路径的中间节点,则$\gamma_{k,u,r,i}=1$;反之,$\gamma_{k,u,r,i}=0$.$\Delta\eta_{i,j}$表示为$\Delta\eta_{i,j}=\dfrac{B(e_{k,u,r}^{\text{v}})}{B(e_{i,j}^{\text{s}})}$.

根据映射成本及功耗的定义可知,两者均与底层网络可用资源及VNR所需资源有关,因而存在一定联系,但两者侧重不同,对其分别进行优化可能存在冲突及矛盾,如某些底层节点、链路可能映射成本较低,但映射所需功耗过高;或某些底层节点、链路映射功耗较低,但负载较重,映射成本较高,因此,考虑到映射成本及功耗的重要性,为实现两者的折中,本文定义VNE代价函数为映射成本及底层网络功耗的加权函数,进而以代价函数最小化为目标确定VNE策略.

3.2 限制条件建模

3.2.1 底层节点及链路容量限制条件

为实现虚拟节点成功映射,需保证底层节点可用资源满足虚拟节点资源需求,即

$$\text{C1:}\sum_{t_0=1}^{t}y_{k,t_0}^{a}\left(1-\sum_{t_1=1}^{t}y_{k,t_1}^{d}\right)\left(\sum_{n_{k,u}^{\text{v}}\in N_k^{\text{v}}}\alpha_{k,u,i}C(n_{k,u}^{\text{v}})\right)\leqslant\hat{C}_t(n_i^{\text{s}}) \tag{14}$$

$$\text{C2:}\sum_{t_0=1}^{t}y_{k,t_0}^{a}\left(1-\sum_{t_1=1}^{t}y_{k,t_1}^{d}\right)\left(\sum_{n_{k,u}^{\text{v}}\in N_k^{\text{v}}}\alpha_{k,u,i}T(n_{k,u}^{\text{v}})\right)\leqslant\hat{T}_t(n_i^{\text{s}}) \tag{15}$$

其中,$\hat{C}_t(n_i^{\text{s}})$及$\hat{T}_t(n_i^{\text{s}})$分别为$t$时隙$n_i^{\text{s}}$的可用CPU容量及TCAM容量.

为实现虚拟链路成功映射,底层链路可用资源需满足虚拟链路资源需求,即

$$\text{C3:}\sum_{t_0=1}^{t}y_{k,t_0}^{a}\left(1-\sum_{t_1=1}^{t}y_{k,t_1}^{d}\right)\left(\sum_{e_{k,u,r}^{\text{v}}\in E_k^{\text{v}}}\beta_{k,u,r,i,j}B(e_{k,u,r}^{\text{v}})\right)\leqslant\hat{B}_t(e_{i,j}^{\text{s}}) \tag{16}$$

其中,$\hat{B}_t(e_{i,j}^{\text{s}})$为$t$时隙$e_{i,j}^{\text{s}}$的可用带宽容量.

3.2.2 虚拟节点映射唯一性限制条件

为确保同一VNR中虚拟节点映射的唯一性,要求VNR中的任一虚拟节点仅可映射至一个底层节点,即

$$\text{C4:}\sum_{n_i^{\text{s}}\in N^{\text{s}}}\alpha_{k,u,i}=1,\forall n_{k,u}^{\text{v}}\in N_k^{\text{v}},\forall k \tag{17}$$

此外,每个底层节点仅可支持同一VNR中的一个虚拟节点映射,即

$$\text{C5:}\sum_{n_{k,u}^{\text{v}}\in N_k^{\text{v}}}\alpha_{k,u,i}\leqslant 1,\forall n_i^{\text{s}}\in N^{\text{s}},\forall k \tag{18}$$

需要说明的是,本文考虑的限制条件C4、C5为VNE相关研究中的常见假设[5~17],未考虑同一VNR的多个虚拟节点映射到一个底层节点的情况.在未来研究中,可考虑放宽这一约束条件,支持同一VNR的多个虚拟节点映射到一个底层节点,即需考虑底层节点与虚拟节点之间的一对多映射.

3.2.3 底层链路流守恒限制条件

虚拟链路映射至底层路径时,需满足流守恒约束,即

$$\text{C6:}\sum_{n_j^{\text{s}}\in H(n_i^{\text{s}})}\beta_{k,u,r,i,j}-\sum_{n_j^{\text{s}}\in H(n_i^{\text{s}})}\beta_{k,u,r,j,i}=\alpha_{k,u,i}-\alpha_{k,r,i} \tag{19}$$

其中,$H(n_i^{\text{s}})$表示与n_i^{s}相连的邻居节点集合.

3.2.4 底层节点及链路状态限制条件

若n_i^{s}处于激活状态,则与n_i^{s}相连的底层链路中至少存在一条处于激活状态的底层链路,即

$$\text{C7:}\sum_{n_j^{\text{s}}\in H(n_i^{\text{s}})}x_{i,j,t}+\sum_{n_j^{\text{s}}\in H(n_i^{\text{s}})}x_{j,i,t}\geqslant 1,\ \text{if}\ x_{i,t}=1 \tag{20}$$

与n_i^{s}相连的所有激活底层链路数量之和不应超过n_i^{s}的度数,即

$$\text{C8:}\sum_{n_j^{\text{s}}\in H(n_i^{\text{s}})}x_{i,j,t}+\sum_{n_j^{\text{s}}\in H(n_i^{\text{s}})}x_{j,i,t}\leqslant\left|H(n_i^{\text{s}})\right| \tag{21}$$

其中,$\left|H(n_i^{\text{s}})\right|$为$n_i^{\text{s}}$的度数.若虚拟节点映射至处于休眠状态的底层节点,则该底层节点需切换至激活状态,即

$$\text{C9:}\ x_{i,t}=\left(\prod_{k\in K}\left(1-\sum_{t_0=1}^{t}y_{k,t_0}^{a}\left(1-\sum_{t_1=1}^{t}y_{k,t_1}^{d}\right)\sum_{n_{k,u}^{\text{v}}\in N_k^{\text{v}}}\alpha_{k,u,i}\right)\right)\oplus 1 \tag{22}$$

其中,$\oplus$为异或运算符.类似地,可得虚拟链路状态限制条件如式(23)所示.

$$\text{C10: } x_{i,j,t} = \left(\prod_{k \in K} \left(1 - \sum_{t_0=1}^{t} y^{a}_{k,t_0} \left(1 - \sum_{t_1=1}^{t} y^{d}_{k,t_1} \right) \right. \right. \left. \left. \sum_{e^{v}_{k,u,r} \in E^{v}_{k}} \sum_{n^{s}_{j} \in H(n^{s}_{i})} \beta_{k,u,r,i,j} \right) \right) \oplus 1, \forall e^{s}_{i,j} \in E^{s}, \forall k \quad (23)$$

3.2.5 中间节点限制条件

为标识承载虚拟链路的底层路径中间节点，$\gamma_{k,u,r,j}$ 应满足：

$$\text{C11: } \sum_{n^{s}_{j} \in H(n^{s}_{i})} \beta_{k,u,r,i,j} - \gamma_{k,u,r,j} = \alpha_{k,u,i} \quad (24)$$

3.2.6 虚拟链路最大可容忍时延限制条件

承载虚拟链路的底层链路传播时延之和应满足虚拟链路最大可容忍传播时延限制，即

$$\text{C12: } \sum_{e^{s}_{i,j} \in E^{s}} \beta_{k,u,r,i,j} D(e^{s}_{i,j}) \leqslant D(e^{v}_{k,u,r}) \quad (25)$$

3.3 优化模型建立

本文以VNE代价函数最小化为目标，建模SDN的VNE问题为给定限制条件的代价函数优化问题，即

$$\min_{\alpha_{k,u,i},\beta_{k,u,r,i,j},\gamma_{k,u,r,i}} \Theta \quad \text{s.t.} \quad \text{C1-C12} \quad (26)$$

4 优化问题求解

由于所建模优化问题(26)中目标函数以及部分约束条件，如C9、C10均具有非线性特性，式(26)为复杂非线性整数优化问题，难以使用典型数学模型求解. 为求解该问题，本文首先应用基于时间窗的批处理映射策略动态处理VNR，继而针对特定时间窗内的VNE问题，将其转换为虚拟节点映射子问题和虚拟链路映射子问题，进而分别进行求解. 本文未来研究中，将考虑采用变量转换、非线性优化等工具对所建模优化问题进行求解.

4.1 基于时间窗的VNR批处理策略

由于在较长时段实现动态VNE存在较大难度，本文提出基于时间窗的VNR批处理策略，即将映射时段离散化为由多个时隙组成的时间窗，进而在特定时间窗内对已到达且未失效的VNR进行集中映射. 该过程持续进行直至VNR失效.

针对特定时间窗内的多个VNR，本节综合考虑VNR所需资源量及生存时间，定义VNR优先级，进而根据优先级依次进行映射. 各VNR对底层资源可能提出不同需求，而由于底层资源的有限性及动态变化特性，资源需求较高的VNR实现成功映射的难度较大，因而需优先保障底层资源的可用性；此外，由于VNR在不同时隙到达及失效，为实现尽可能多的VNR的成功映射，还需考虑VNR的生存时间，如对于生存时间较短的VNR赋予较高优先级. 基于此，本文建模 VNR_k 的优先级 $\xi(\text{VNR}_k)$ 为

$$\xi(\text{VNR}_k) = \frac{R(\text{VNR}_k)}{\sigma_k} \quad (27)$$

其中，$R(\text{VNR}_k)$ 为 VNR_k 所需底层资源总量，σ_k 为 VNR_k 的生存时间. $R(\text{VNR}_k)$ 定义为 VNR_k 的虚拟节点及链路资源需求量之和，如式(28)所示.

$$R(\text{VNR}_k) = \omega_{\text{C}} \sum_{n^{v}_{k,u} \in N^{v}_{k}} C(n^{v}_{k,u}) + \omega_{\text{T}} \sum_{n^{v}_{k,u} \in N^{v}_{k}} T(n^{v}_{k,u}) + \omega_{\text{B}} \sum_{e^{v}_{k,u,r} \in E^{v}_{k}} B(e^{v}_{k,u,r}) \quad (28)$$

其中，ω_{C}、ω_{T}、ω_{B} 分别表示CPU资源、TCAM资源及带宽资源的权值系数；σ_k 定义为 $\sigma_k = t^{d}_{k} - t^{a}_{k}$.

由式(27)可知，若 $R(\text{VNR}_k)$ 较大或 σ_k 较小，则 VNR_k 的优先级较高，因而可优先映射，从而有效实现底层资源的高效利用，提高VNR成功映射概率. 针对某时间窗内的多个VNR，将根据VNR的优先级依次进行映射；对于给定 VNR_k 的映射问题，将其转化为两个子问题，即虚拟节点映射子问题和虚拟链路映射子问题，分别进行求解.

4.2 虚拟节点映射子问题求解

本小节首先建模虚拟节点映射子问题，并采用贪婪算法进行求解.

4.2.1 子问题建模

本小节建模给定时间窗内的虚拟节点映射子问题，令 t 表示当前时间窗的截止时隙，$\Theta^{n}_{k,t}$ 表示 VNR_k 的虚拟节点映射代价，建模 $\Theta^{n}_{k,t}$ 如下：

$$\Theta^{n}_{k,t} = \mu_Q Q^{n}_{k,t} + \mu_P P^{n}_{k,t} \quad (29)$$

VNR_k 的虚拟节点映射子问题建模为

$$\min_{\alpha_{k,u,i}} \Theta^{n}_{k,t} \quad \text{s.t. C1, C4, C5, C7, C8, C12} \quad (30)$$

为求解上述优化问题，本文提出一种基于贪婪算法的虚拟节点映射子算法，首先分别定义底层节点及虚拟节点的度量值，进而根据节点度量值，依次实现虚拟节点与底层节点间的匹配映射.

4.2.2 底层节点度量值

将 $n^{v}_{k,u}$ 映射至 n^{s}_{i} 时，需综合考虑 n^{s}_{i} 的属性及与其相连的底层链路属性，因此，本小节提出一种底层节点度量方法. 定义 n^{s}_{i} 的度量值 $\psi^{s}_{t}(n^{s}_{i})$ 为 n^{s}_{i} 的映射能力与其所相连链路的平均映射能力之和，即

$$\psi^{s}_{t}(n^{s}_{i}) = \varphi_t(n^{s}_{i}) + \frac{1}{\left| H(n^{s}_{i}) \right|} \sum_{n^{s}_{j} \in H(n^{s}_{i})} \varphi_t(e^{s}_{i,j}) \quad (31)$$

其中，$\varphi_t(n^{s}_{i})$ 和 $\varphi_t(e^{s}_{i,j})$ 分别表示 t 时隙 n^{s}_{i} 的映射能力及 $e^{s}_{i,j}$ 的映射能力. $\varphi_t(n^{s}_{i})$ 表示为

$$\varphi_t(n^{s}_{i}) = \mu_Q q_t(n^{s}_{i}) + \mu_P p_t(n^{s}_{i}) \quad (32)$$

其中，$q_t(n^{s}_{i})$ 和 $p_t(n^{s}_{i})$ 分别表示 t 时隙 n^{s}_{i} 的资源度量值及

功耗度量值. 定义$q_t(n_i^{\mathrm{s}})$如式(33)所示.

$$q_t(n_i^{\mathrm{s}})=\lambda^{\mathrm{C}}\hat{C}_t(n_i^{\mathrm{s}})(1+\exp(-\delta^{\mathrm{C}}\eta_{i,t}^{\mathrm{C}}))+\lambda^{\mathrm{T}}\hat{T}_t(n_i^{\mathrm{s}})(1+\exp(-\delta^{\mathrm{T}}\eta_{i,t}^{\mathrm{T}})) \tag{33}$$

$p_t(n_i^{\mathrm{s}})=x_{i,t}(P_i^{\max}-P_t(n_i^{\mathrm{s}}))$,其中,$p_t(n_i^{\mathrm{s}})$表示$t$时隙$n_i^{\mathrm{s}}$的功耗值. 式(31)中,$\varphi_t(e_{i,j}^{\mathrm{s}})$表示为

$$\varphi_t(e_{i,j}^{\mathrm{s}})=\lambda^{\mathrm{B}}\hat{B}_t(e_{i,j}^{\mathrm{s}})(1+\exp(-\delta^{\mathrm{B}}\eta_{i,j,t}^{\mathrm{B}})) \tag{34}$$

4.2.3 虚拟节点度量值

考虑到底层资源的有限性,对底层网络资源需求较大的虚拟节点更难映射成功,应优先映射该类虚拟节点以提升VNE性能. 本文定义$n_{k,u}^{\mathrm{v}}$的度量值$\psi^{\mathrm{v}}(n_{k,u}^{\mathrm{v}})$如式(35)所示,其中,$\left|H(n_{k,u}^{\mathrm{v}})\right|$为$n_{k,u}^{\mathrm{v}}$的度数.

$$\psi^{\mathrm{v}}(n_{k,u}^{\mathrm{v}})=(C(n_{k,u}^{\mathrm{v}})+T(n_{k,u}^{\mathrm{v}}))+\frac{1}{\left|H(n_{k,u}^{\mathrm{v}})\right|}\sum_{n_{k,r}^{\mathrm{v}}\in H(n_{k,u}^{\mathrm{v}})}B(e_{k,u,r}^{\mathrm{v}}) \tag{35}$$

4.2.4 基于贪婪算法的节点映射策略

本小节提出基于贪婪算法的节点映射策略,所提策略步骤如下.

步骤1 根据式(35)计算VNR_k中虚拟节点的度量值,令$n_{k,u^*}^{\mathrm{v}}=\arg\max\limits_{u}(\psi^{\mathrm{v}}(n_{k,u}^{\mathrm{v}}))$为具有最高度量值的虚拟节点;以$n_{k,u^*}^{\mathrm{v}}$为根节点执行广度优先搜索算法,并根据距离$n_{k,u^*}^{\mathrm{v}}$的跳数将剩余虚拟节点进行划分,并对同一集合中的虚拟节点按$\psi^{\mathrm{v}}(n_{k,u}^{\mathrm{v}})$值由大到小进行重排序,从而确定$\mathrm{VNR}_k$的虚拟节点映射序列$\bar{N}_k^{\mathrm{v}}$.

步骤2 根据式(31)计算底层节点的度量值,记$n_{i^*}^{\mathrm{s}}=\arg\max\limits_{i}(\psi_t^{\mathrm{s}}(n_i^{\mathrm{s}}))$为具有最高度量值的底层节点.

步骤3 将n_{k,u^*}^{v}映射至$n_{i^*}^{\mathrm{s}}$,更新底层节点资源,在$\bar{N}_k^{\mathrm{v}}$中删除n_{k,u^*}^{v}.

步骤4 对底层网络中未被映射的底层节点重新进行度量,令$\kappa_t(n_j^{\mathrm{s}})$表示$n_j^{\mathrm{s}}$的度量值,定义为

$$\kappa_t(n_j^{\mathrm{s}})=\frac{1}{\varphi_t(n_j^{\mathrm{s}})}\sum_{n_h^{\mathrm{s}}\in N_{\mathrm{sel}}^{\mathrm{s}}}L_t(d(n_j^{\mathrm{s}},n_h^{\mathrm{s}})) \tag{36}$$

其中,$d(n_j^{\mathrm{s}},n_h^{\mathrm{s}})$表示$n_j^{\mathrm{s}}$与$n_h^{\mathrm{s}}$之间的最短底层路径,$L_t(d(n_j^{\mathrm{s}},n_h^{\mathrm{s}}))$表示$t$时隙$d(n_j^{\mathrm{s}},n_h^{\mathrm{s}})$长度.

步骤5 选择$\bar{N}_k^{\mathrm{v}}$中具有最小跳数且最大度量值的$n_{k,r}^{\mathrm{v}}$映射至具有最小$\kappa_t(n_j^{\mathrm{s}})$值的$n_j^{\mathrm{s}}$,更新底层节点资源,在$\bar{N}_k^{\mathrm{v}}$中删除$n_{k,r}^{\mathrm{v}}$.

步骤6 重复步骤4、步骤5,直至所有虚拟节点完成映射或已无满足资源需求的底层节点.

4.3 虚拟链路映射子问题求解

若VNR_k的虚拟节点映射已完成,即已确定虚拟节点映射策略,VNR_k虚拟链路映射问题即为给定需映射底层路径的源节点与目的节点,确定端到端路由策略.

4.3.1 子问题建模

假定VNR_k中$e_{k,u,r}^{\mathrm{v}}$的端节点$n_{k,u}^{\mathrm{v}}$及$n_{k,r}^{\mathrm{v}}$已映射至n_i^{s}及n_j^{s},即$\alpha_{k,u,i}^{*}=1$及$\alpha_{k,r,j}^{*}=1$,$e_{k,u,r}^{\mathrm{v}}$映射代价函数$\Theta_t^{(k,u,r)}$建模为

$$\Theta_t^{(k,u,r)}=\sum_{e_{i,j}^{\mathrm{s}}\in E^{\mathrm{s}}}\beta_{k,u,r,i,j}\varpi_t^{(k,u,r,i,j)}+\sum_{n_i^{\mathrm{s}}\in N^{\mathrm{s}}}\gamma_{k,u,r,i}\varpi_t^{(k,u,r,i)} \tag{37}$$

其中,$\varpi_t^{(k,u,r,i,j)}$和$\varpi_t^{(k,u,r,i)}$分别表示$e_{k,u,r}^{\mathrm{v}}$映射至$e_{i,j}^{\mathrm{s}}$的链路代价及中间节点激活功耗. $\varpi_t^{(k,u,r,i,j)}$的定义如式(38)所示,$\varpi_t^{(k,u,r,i)}$表示为$\varpi_t^{(k,u,r,i)}=\mu_P(1-x_{i,t})P_i^{\mathrm{on}}$.

$$\varpi_t^{(k,u,r,i,j)}=\mu_Q\Gamma_t^{\mathrm{B}}(e_{i,j}^{\mathrm{s}})B(e_{k,u,r}^{\mathrm{v}})+\mu_P((1-x_{i,j,t})P_{i,j}^{\mathrm{on}}+(P_{i,j}^{\max}-P_{i,j}^{\mathrm{on}})\Delta\eta_{i,j}) \tag{38}$$

$e_{k,u,r}^{\mathrm{v}}$映射子问题表示为

$$\min_{\beta_{k,u,r,i,j},\gamma_{k,u,r,i}}\ \Theta_t^{(k,u,r)}\quad \text{s.t.}\quad \mathrm{C2,C3,C6,C9\text{-}C11} \tag{39}$$

为求解上述优化问题,本文提出一种基于K最短路径算法的虚拟链路映射策略.

4.3.2 基于K最短路径算法的链路映射策略

为求解式(39)建模的优化问题,构建权重图$\bar{G}^{\mathrm{s}}=(\bar{N}^{\mathrm{s}},\bar{E}^{\mathrm{s}},\bar{W}^{\mathrm{s}})$,其中,$\bar{W}^{\mathrm{s}}=\{w_{i,j}^{\mathrm{s}}\}$表示底层链路权重集合,$w_{i,j}^{\mathrm{s}}$为$e_{i,j}^{\mathrm{s}}$的权重,定义为$e_{k,u,r}^{\mathrm{v}}$映射至$e_{i,j}^{\mathrm{s}}$的链路代价,即$w_{i,j}^{\mathrm{s}}=\varpi_t^{(k,u,r,i,j)}$.

为实现计算复杂度和映射性能的折中,本文采用K最短路径算法确定$e_{k,u,r}^{\mathrm{v}}$映射的K条候选底层路径集合. 针对各候选底层路径,进一步考虑中间节点激活功耗,选择最优底层路径.

基于K最短路径算法的虚拟链路映射步骤如下.

步骤1 根据虚拟链路带宽容量需求构造映射序列$\bar{E}_k^{\mathrm{v}}$,对具有高带宽资源需求的虚拟链路优先映射.

步骤2 对待映射的$e_{k,u,r}^{\mathrm{v}}$,构建权重图$\bar{G}^{\mathrm{s}}=(\bar{N}^{\mathrm{s}},\bar{E}^{\mathrm{s}},\bar{W}^{\mathrm{s}})$,其中,$\bar{E}^{\mathrm{s}}$表示满足$e_{k,u,r}^{\mathrm{v}}$带宽及可容忍时延限制的底层链路集合,计算底层链路权重集合$\bar{W}^{\mathrm{s}}$.

步骤3 在$\bar{G}^{\mathrm{s}}$中执行K最短路算法,获得K条候选底层路径,选择满足可容忍时延限制的候选底层路径集合$\bar{P}_{k,u,r}=\{\bar{p}_{k,u,r}^{(1)},\bar{p}_{k,u,r}^{(2)},\ldots,\bar{p}_{k,u,r}^{(\bar{L})}\}$.

步骤4 若$\bar{P}_{k,u,r}$为空,则算法终止;否则,计算$\bar{P}_{k,u,r}$中各底层路径的映射代价值,令$\Theta_t^{(k,u,r,l)}$为$\bar{p}_{k,u,r}^{(l)}$的映射代价值.

步骤5 比较$\bar{L}$条候选底层路径的映射代价值,选择具有最小映射代价值的底层路径,即$(\beta_{k,u,r,i,j}^{*},\gamma_{k,u,r,i}^{*})=\arg\min\limits_{l}(\Theta_t^{(k,u,r,l)})$,更新底层链路资源,并在$\bar{E}_k^{\mathrm{v}}$中删除$e_{k,u,r}^{\mathrm{v}}$.

步骤6 重复步骤2至步骤5,直至$\bar{E}_k^{\mathrm{v}}$中所有虚拟链路完成映射或已无满足资源需求的底层链路.

4.4　算法复杂度

本小节对本文所提的虚拟节点及虚拟链路映射子问题的时间复杂度进行分析.

4.4.1　虚拟节点映射算法

为求解虚拟节点映射子问题,本文提出一种基于贪婪算法的虚拟节点映射子算法,首先分别定义底层节点及虚拟节点的度量值,进而根据节点度量值,依次实现虚拟节点与底层节点间的匹配映射. 对于虚拟网络请求 VNR_k,虚拟节点映射算法涉及对各虚拟节点及底层节点的度量,其中,虚拟节点度量复杂度为 $O(M_k)$. 由于每完成一次虚拟节点映射,均需对底层节点状态进行更新,并进行底层节点重新度量,故所需计算量为 $M(M+1)/2-(M-M_k)(M-M_k+1)/2$,计算复杂度为 $O(M_kM^2)$,因此,对虚拟网络请求 VNR_k,其虚拟节点映射算法复杂度为 $O(M_kM^2)$.

4.4.2　虚拟链路映射算法

为求解虚拟链路映射子问题,本文提出一种基于K最短路径算法的虚拟链路映射子算法,具体而言,对于每条虚拟链路,均执行K最短路径算法,获得 K 条候选底层路径,并从中选择满足可容忍时延限制的候选底层路径. 令 L 为满足可容忍时延限制的候选底层路径数目,因底层节点数目为 M,执行一次K最短路径算法的复杂度为 $O(M(L+M\log M))$. 令虚拟网络请求 VNR_k 中虚拟链路数目为 L_k,则执行 VNR_k 的虚拟链路映射所需计算复杂度为 $O(KML_k(L+M\log M))$.

5　算法仿真及性能分析

本小节应用MATLAB仿真软件对所提算法性能进行仿真评估.

5.1　仿真场景及参数设置

本文算法仿真阶段考虑底层网络中各节点随机分布在 $1000\times1000\mathrm{km}^2$ 的区域内,底层节点数目分别设置为30和40个. 本文实验环节参考VNE相关研究常用方法及参数设置[5~10],将底层节点的初始CPU容量、初始TCAM容量和底层链路的初始带宽容量均设定为归一化参数,假设均服从均匀分布 $U(50,100)$. 类似地,对底层节点及链路的功耗进行归一化设置,假设底层节点基础功耗及满载功耗分别服从均匀分布 $U(150,200)$ 及 $U(600,800)$,底层链路的基础功耗及满载功耗分别服从均匀分布 $U(100,150)$ 及 $U(300,400)$. 根据映射成本及功耗所对应的取值范围,为实现加权后的两者取值接近,将相应权重因子 μ_Q 和 μ_P 分别设置为4和1.

假定VNR到达服从泊松过程,每个VNR的生存时间服从均值为250个时间单元的指数分布. VNR的虚拟节点随机分布在 $1000\times1000\mathrm{km}^2$ 的区域内. 每个VNR的虚拟节点数目服从均匀分布 $U(2,8)$,虚拟节点的CPU容量需求、TCAM容量需求和虚拟链路的带宽容量需求均服从均匀分布 $U(1,20)$. 仿真时间设置为100个时间窗,算法结果为500次独立仿真结果的平均值. 对比起见,仿真图中给出了本文所提算法与文献[10]所提算法的性能.

5.2　仿真结果分析

图2与图3中,VNR到达数目服从均值为4的泊松过程. 图2对比了不同底层节点数目对应VNR映射代价随仿真时间的变化关系,其中,映射代价为特定仿真时间前所有底层网络映射代价之和. 从图中可以看出,随着仿真时间的增加,本文及文献[10]所提算法对应的VNR映射代价均有所增大. 与文献[10]所提算法对比,本文所提算法可获得较低的映射代价.

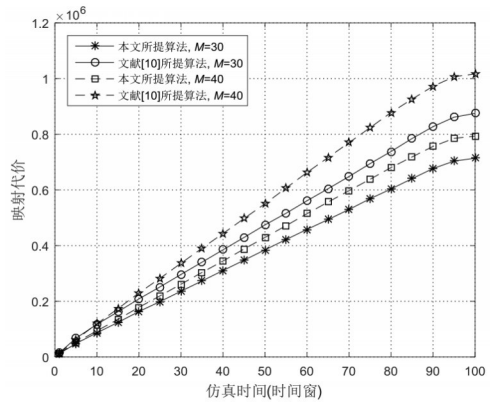


图2　VNR映射代价与仿真时间关系图(不同底层节点数目)

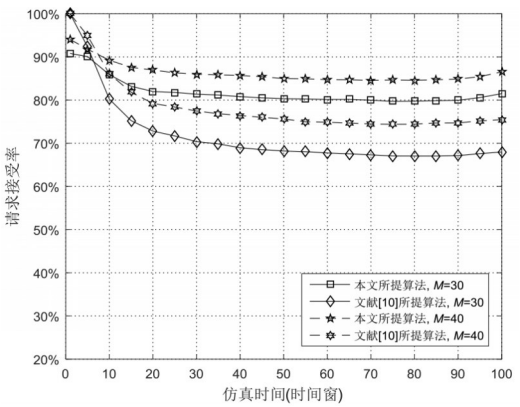


图3　请求接受率与仿真时间关系图(不同底层节点数目)

图3给出了不同底层节点数目对应VNR接受率,其中,VNR接受率为特定仿真时间前VNR接受率之和. 由图可知,仿真初始阶段两种算法对应请求接受率均较高,这是因为仿真开始阶段底层网络可用资源较多. 随着仿真时间增加,两种算法对应的请求接受率均有所下降,这是因为随着VNR的持续到达,底层网络可

用资源量减少,请求接受率相应降低. 与文献[10]相比,本文所提算法的请求接受率较高,原因是本文所提算法在VNE过程中联合考虑网络可用资源及VNR动态特性,在节点及链路映射阶段考虑了多种因素以及当前映射策略对后续映射性能的影响,因而较对比算法具有更优性能.

图4与图5中,VNR到达数目分别服从均值为5和8的泊松过程. 图4给出了底层网络节点数目为30,VNR平均到达率分别为5和8时,映射代价与仿真时间的关系图. 从图中可以看出,随着仿真时间的增加,VNE代价增大. 对比文献[10]所提算法,本文算法对应的映射代价较低.

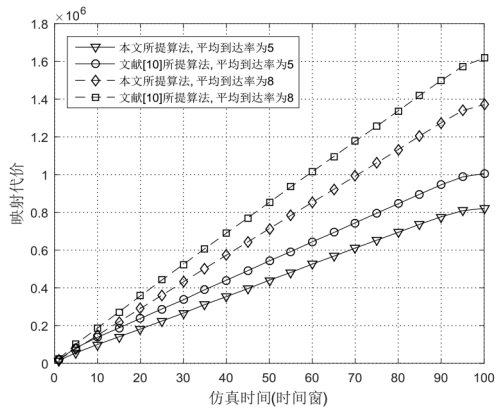


图4 VNR映射代价与仿真时间关系图(不同请求到达率)

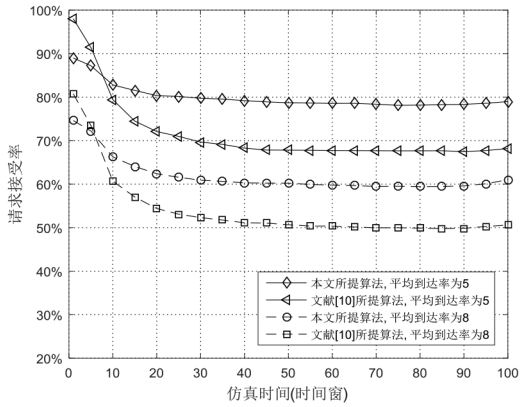


图5 请求接受率与仿真时间关系图(不同请求到达率)

图5给出了底层网络节点数目为30,VNR平均到达率为5和8时,VNR接受率与仿真时间的关系图. 从图中可以看出,平均到达率增加时,请求接受率降低. 这是因为到达率的增加导致时间窗内到达的VNR增多,从而请求接受率下降. 与文献[10]相比,本文所提算法具有更高的请求接受率.

图6给出了不同底层节点数目对应VNR映射代价与平均到达率的关系图. 图6中,为评估映射代价与VNR到达率的关系,选择某个时间窗,如第10个时间窗,评估该时间窗内映射代价函数. 由图可知,随着VNR平均到达率的增加,底层网络映射代价相应增加,这是因为随着更多VNR的到达,底层网络的可用资源量逐渐减少,映射成本增加,导致映射代价相应增加. 与文献[10]对比,本文所提算法对应的映射代价较低.

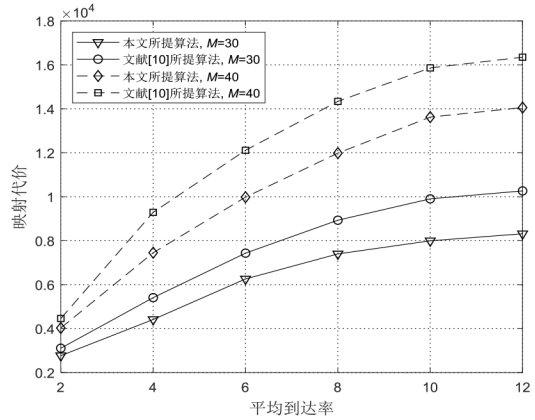


图6 映射代价与平均到达率关系图(不同底层节点数目)

图7给出了不同底层节点数目对应VNR请求接受率与VNR平均到达率的关系图. 与图6类似,图7中选取第10个时间窗,绘制该时间窗内VNR请求接受率. 由图可知,随着VNR平均到达率增加,请求接受率相应降低,这是因为随着更多VNR的到达,VNR映射对于底层网络资源的竞争加剧,底层网络可用资源量减少,导致请求接受率降低. 对比文献[10]所提算法,本文所提算法具有较高请求接受率.

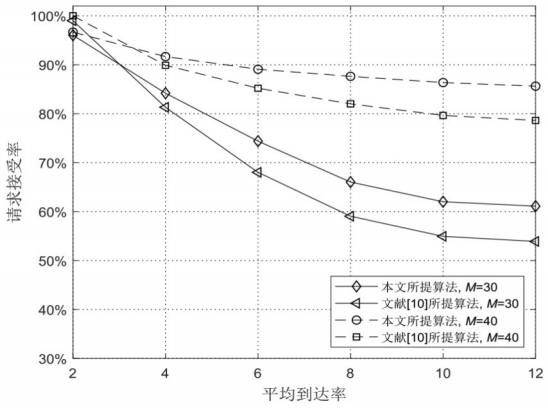


图7 请求接受率与平均到达率关系图(不同底层节点数目)

图8给出了VNR平均到达数目分别为2和6时,底层网络负载均衡度与仿真时间关系图. 图中,底层网络负载均衡度定义为所有底层节点及链路负载率的方差[10]. 由图可见,随着仿真时间的变化,底层网络负载均衡度动态变化. 对于较大的VNR平均到达数目,底

层网络负载均衡度较高,这是因为更多VNR导致底层网络负载增加,负载率的方差相应有所增加. 与文献[10]所提算法进行对比,本文所提算法对应的底层网络负载均衡度较低,主要原因是本文所提算法在对映射成本及映射功耗进行评估时,均考虑底层网络负载特性,映射代价的优化导致底层网络资源的相对均衡.

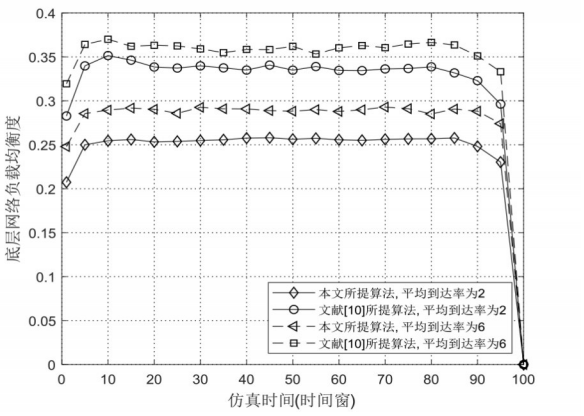


图8　底层网络负载均衡度与仿真时间关系图(不同平均到达率)

6　结束语

针对多个VNR动态到达的网络场景,联合考虑VNE成本和功耗,本文提出一种基于成本及功耗联合优化的SDN的VNE策略. 综合考虑底层资源容量、流守恒及可容忍时延等限制条件,建立基于代价函数最小化的VNE模型. 应用基于时间窗的批处理映射策略动态处理VNR,继而针对特定时间窗内的VNR,将其转换为虚拟节点映射子问题和虚拟链路映射子问题,并提出启发式算法对两个子问题分别进行求解. 仿真验证所提算法可实现VNE代价函数的优化,验证了所提算法的有效性.

参考文献

[1] Kreutz D, Ramos F M V, Verissimo P E, et al. Software-defined networking: a comprehensive survey[J]. Proceedings of the IEEE, 2015, 103(1): 17 – 76.

[2] Khan A, Zugenmaier A, Jurca D, et al. Network virtualization: a hypervisor for the Internet[J]. IEEE Communications Magazine, 2012, 50(1): 136 – 143.

[3] Chowdhury M, Rahman M R, Boutaba R. ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping[J]. IEEE/ACM Transactions on Networking, 2012, 20(1): 206 – 219.

[4] Fischer A, Botero J F, Beck M T, et al. Virtual network embedding: a survey[J]. IEEE Communications Surveys and Tutorials, 2013, 15(4): 1888 – 1906.

[5] 刘焕淋, 胡浩, 陈勇, 等. 联合能耗与负载均衡的虚拟网络映射方法[J]. 电子学报, 2019, 47(12): 2488 – 2494.
Liu H L, Hu H, Chen Y, et al. Joint power consumption and load balancing algorithm for virtual optical network embedding[J]. Acta Electronica Sinica, 2019, 47(12): 2488 – 2494. (in Chinese)

[6] Zhang P Y, Yao H P, Liu Y J. Virtual network embedding based on computing, network, and storage resource constraints[J]. IEEE Internet of Things Journal, 2018, 5(5): 3298 – 3304.

[7] Liu X, Zhang Z B, Li X M, et al. Optimal virtual network embedding based on artificial bee colony[J]. EURASIP Journal on Wireless Communications and Networking, 2016, 2016(1): 1 – 9.

[8] Yan Z X, Ge J G, Wu Y L, et al. Automatic virtual network embedding: a deep reinforcement learning approach with graph convolutional networks[J]. IEEE Journal on Selected Areas in Communications, 2020, 38(6): 1040 – 1057.

[9] Beck M T, Fischer A, Botero J F, et al. Distributed and scalable embedding of virtual networks[J]. Journal of Network and Computer Applications, 2015, 56: 124 – 136.

[10] Zhang P Y. Incorporating energy and load balance into virtual network embedding process[J]. Computer Communication, 2018, 129: 80 – 88.

[11] Wei W T, Gu H X, Wang K, et al. Improving cloud-based IoT through virtual network embedding in elastic optical inter-DC networks[J]. IEEE Internet of Things Journal, 2019, 6 (1): 986 – 996.

[12] Beck M T, Fischer A, Botero J F, et al. Distributed and scalable embedding of virtual networks[J]. Journal of Network and Computer Applications, 2015, 56: 124 – 136.

[13] Zong Y, Ou Y N, Hammad A, et al. Location-aware energy efficient virtual network embedding in software-defined optical data center networks[J]. IEEE/OSA Journal of Optical Communications and Networking, 2018, 10(7): 58 – 70.

[14] Nonde L, Ei-Gorashi T E H, Eimirghani J M H. Energy efficient virtual network embedding for cloud networks[J]. Journal of Lightwave Technology, 2015, 33(9): 1828 – 1849.

[15] Zhang Z B, Su S, Shuang K, et al. Energy aware virtual network migration[A]. IEEE Global Communications Conference[C]. Washington, DC, USA: IEEE, 2016. 1 – 6.

[16] Alaluna M, Neves N, Ramos F M V. Elastic network virtualization[A]. IEEE Conference on Computer Communications[C]. Toronto, Canada: IEEE, 2020. 814 – 823.

[17] Li J L, Zhang N, Ye Q, et al. Joint resource allocation and online virtual network embedding for 5G networks[A]. IEEE Global Communications Conference[C]. Singapore, Singapore: IEEE, 2017. 1 – 6.

[18] Hejja K, Hesselbach X. Online power aware coordinated virtual network embedding with 5G delay constraint[J]. Journal of Network and Computer Applications, 2018, 124: 121 – 136.

[19] Li Z F, Lu Z B, Deng S H, et al. A self-adaptive virtual network embedding algorithm based on software-defined networks[J]. IEEE Transactions on Network and Service Management, 2019, 16(1): 362 – 373.

[20] Cao H T, Zhu Y X, Zheng G, et al. A novel optimal mapping algorithm with less computational complexity for virtual network embedding[J]. IEEE Transactions on Network and Service Management, 2018, 15(1): 356 – 371.

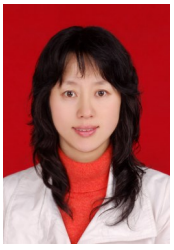
[21] Thakkar H K, Dehury C K, Sahoo P K. MUVINE: Multi-stage virtual network embedding in cloud data centers using reinforcement learning-based data predictions[J]. IEEE Journal on Selected Areas in Communications, 2020, 38 (6): 1058 – 1074.

[22] Lu M L, Gu Y, Xie D L. A dynamic and collaborative multi-layer virtual network embedding algorithm in SDN based on reinforcement learning[J]. IEEE Transactions on Network and Service Management, 2020, 17(4): 2305 – 2317.

[23] Bishop C M. Pattern Recognition and Machine Learning [M]. New York, USA: Springer, 2006.

[24] Shenker S. Fundamental design issues for the future Internet[J]. IEEE Journal on Selected Areas in Communications, 1995, 13(7): 1176 – 1188.

作者简介



柴 蓉 女,1974 出生.重庆邮电大学教授、博士生导师,主要研究方向为移动通信、卫星通信、软件定义网络、无线资源管理及移动性管理技术等. E-mail:chairong@cqupt.edu.cn

谢德胜 男,1994 年出生.重庆邮电大学硕士研究生,主要研究方向为移动通信技术、软件定义网络及网络虚拟化等.

陈前斌 男,1967 年出生.重庆邮电大学副校长、教授、博士生导师,主要研究方向为个人通信、移动通信、多媒体信息处理与传输等.