文章编号:1007-5321(2018)03-0075-06 DOI:10.13190/j.jbupt.2017-213

SDN-ScaSVNE:可伸缩的 SDN 生存性虚拟网络映射算法

卢美莲, 顾 云, 刘 通
(北京邮电大学 网络与交换技术国家重点实验室, 北京 100876)

摘要:现有的生存性虚拟网络映射算法无法直接应用于软件定义网络(SDN),而且大多数算法仅通过扩展虚拟网络提供主动保护策略,性能较差. 对此,提出了一种基于剩余网络资源可伸缩备份虚拟资源的 SDN 生存性虚拟网络映射算法,仅为扩展虚拟网络中满足备份约束的节点和链路提供备份资源,并利用剩余网络资源和备份资源共同完成故障恢复. 仿真结果表明,算法在拥有较高收益/成本比值的前提下,可有效提高请求接受率和故障恢复成功率.

关 键 词:生存性虚拟网络映射;软件定义网络;可伸缩备份资源
中图分类号:TP393.0 **文献标志码**:A

SDN-ScaSVNE: Scalable Survivable Virtual Network Embedding Algorithm in SDN

LU Mei-lian, GU Yun, LIU Tong
(State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China)

Abstract: Existing survivable virtual network embedding algorithms cannot apply to software defined network (SDN) directly, and most of them only use active protection strategy by augmenting virtual networks, which leads to poor performance. For the problems above, a survivable virtual network embedding algorithm based on residual network resources is proposed for scalable backup virtual resources in SDN. It provides backup resources only for nodes and links that satisfy the backup constraints in the augmented virtual networks, and combines the remaining network resources with backup resources to complete failure recovery. Simulation results show that the algorithm can effectively improve the acceptance rate of virtual network requests and the success rate of failure recovery under the premise of high revenue/cost ratio.

Key words: survivable virtual network embedding; software-defined network; scalable backup resource

虚拟网络映射(VNE, virtual network embedding)实现了虚拟网络请求(VNR, virtual network request)到底层物理网络(SN, substrate network)的映射. 生存性虚拟网络映射(SVNE, survivable virtual network embedding)是更加严格的 VNE,保障被映射成功的 VNR 在 SN 发生故障时依然能够正常提供服务.

软件定义网络(SDN, software defined network)是一种新型网络架构,通过分离控制平面和数据平面实现对网络流量的灵活控制. 流表是 SDN 交换机进行数据包转发处理的根本依据,直接影响 SDN 数据转发的效率. 所以相较传统网络,SDN 中的 SVNE 需要额外考虑物理节点的流表空间资源[1].

现有针对 SVNE 的研究主要分为两类. Yu

收稿日期:2017-10-22
基金项目:国家自然科学基金项目(61471060)
作者简介:卢美莲(1967—),女,副教授,E-mail:mllu@bupt.edu.cn.

等[2]提出了基于扩展虚拟网络请求(A-VNR,augmented VNR)的主动保护策略,即在映射前为VNR置备冗余的节点和链路. Hu等[3]则进一步加入了节点的位置约束. 但是这些主动保护策略增加了VNR资源需求,容易导致映射失败,且仅使用单一备份资源进行故障恢复,故障恢复成功率较低. Bo等[4]提出的被动恢复策略则不需要预置备份资源,其基于当前网络剩余资源进行故障恢复. Xiao等[5]将物理资源划分为主要和次要资源,分别用于VNR映射和故障恢复. 这些被动保护策略虽然没有增加额外的资源消耗,但是增加了故障恢复阶段的耗时,效率较低.

为改善这两类策略会消耗更多底层网络资源以及故障恢复资源单一问题,针对SDN提出了一种基于网络剩余资源可伸缩备份虚拟资源的SVNE算法——SDN-ScaSVNE. 当网络剩余资源充足时才对有需要的虚拟节点和链路提供备份,故障恢复阶段会优先使用备份资源,否则引入被动恢复策略重映射因故障而失效的虚拟节点和链路.

1 网络模型和问题描述

1.1 网络模型

图1所示为SVNE的网络模型. 其中SN被抽象为无向图 $G^S=(N^S,E^S,C^S)$,N^S 和 E^S 分别为物理节点和链路集合,节点资源包括CPU计算资源 $c(s)$ 和流表资源 $h(s)$,节点的位置设为 $\mathrm{loc}(s)$,链路资源包括带宽资源 $b(l_{mn})$. C^S 为约束条件,包括节点位置约束、CPU和流表资源约束以及链路带宽资源约束.

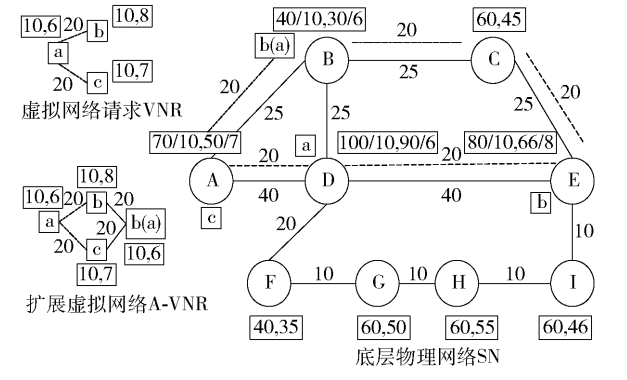

图1 SVNE网络模型

VNR被抽象为无向图 $G^V=(N^V,E^V,C^V)$,N^V 和 E^V 分别为虚拟节点和链路集合. C^V 为约束条件,包括节点位置 $\mathrm{loc}(v)$、请求的CPU资源 $c(v)$ 和流表资源 $h(v)$ 以及链路带宽资源 $b(e_i^V)$. A-VNR表示采用 k 冗余备份策略[2]得到的扩展VNR,k 为VNR中重要节点个数,每个重要节点及其关联链路都会预置备份. 运行时间 T 内所有故障节点集合为

$$F^N=\bigcup_{i=0}^{T}\{f_s\} \tag{1}$$

其中 f_s 表示某个时刻发生故障的物理节点.

1.2 问题描述

SVNE通过提高请求接受率和故障恢复成功率,最终提高SN的网络运营收益. SDN中节点资源模型不同于传统网络,因此单个VNR收益重定义为

$$R(G^V)=\zeta\sum_{v\in N^V}c(v)+\lambda\sum_{v\in N^V}h(v)+\pi\sum_{e_i^V\in E^V}b(e_i^V) \tag{2}$$

其中:ζ、λ 和 π 分别为虚拟节点CPU、流表以及虚拟链路带宽的映射单价,SN运营商可根据实际情况调节. 这里假定成本单价均为1个单位,映射单价均为10个单位,即每种资源可获得10倍利润.

在VNR运行时间内若有物理节点发生故障,则故障恢复失败所导致的赔偿金定义为

$$P(f_s,G^V)=\begin{cases}\rho R(G^V), & \text{故障恢复失败}\\ 0, & \text{未受}f_s\text{影响或故障恢复成功}\end{cases} \tag{3}$$

其中 ρ 为赔偿金和收益的可调比例,设为0.2.

根据式(2)和式(3)得出网络运营总收益为

$$\mathrm{Profit}(T)=\sum_{G^V\in S(T)}\{R(G^V)-\sum_{f_s\in F^N}P(f_s,G^V)\} \tag{4}$$

其中 $S(T)$ 和 F^N 表示在SN提供服务的总时间 T 内被成功映射的VNR集合以及故障节点集合.

网络开销定义为SN为VNR分配的资源总和

$$C(G^V)=\Big[\sum_{e_i^V\in E^V}\sum_{l_{mn}\in E^S}b_{mn}^i+b_b\Big]+\Big[\sum_{v\in N^V}c(v)+c_b\Big]+\Big[\sum_{v\in N^V}h(v)+h_b\Big] \tag{5}$$

其中:b_{mn}^i 为物理链路 l_{mn} 为第 i 条虚拟链路 e_i^V 分配的带宽资源,c_b 和 h_b 分别为备份的CPU和流表资源,b_b 为带宽备份资源. 由式(5)得出SN总开销为

$$\mathrm{Cost}(T)=\sum_{G^V\in S(T)}C(G^V) \tag{6}$$

2 用于VNE求解的线性规划模型

Chowdhury等[6]通过协同节点和链路的两阶段

映射优化了 VNE 问题的求解. 首先根据式(7)的位置约束得到每个虚拟节点的可映射物理节点集合：

$$\phi(v)=\{s\in N^{S}\mid \mathrm{Dis}(\mathrm{loc}(v),\mathrm{loc}(s))\leqslant r^{v}\} \quad (7)$$

其中：$\mathrm{Dis}(\mathrm{loc}(v),\mathrm{loc}(s))$表示虚拟节点 v 的目标映射位置与物理节点 s 的实际位置之间的距离；r^{v} 表示虚拟节点 v 能够接受的最大 Dis 值.

这样 SN 中的节点可以被看成划分在了多个簇中，为每个簇添加一个节点和多条链路，分别称之为元节点和元链路. 每个元节点都代表了一个虚拟节点，它与相应簇 $\phi(v)$ 中的每个物理节点都通过一条带宽资源无穷大的元链路相连，最终得到了扩展的 SN 抽象图 $G^{S'}=(N^{S'},E^{S'})$. 因此 VNE 问题转化为了混合整数多商品流问题，VNR 中的每条虚拟链路即为一条商品流，起点和终点是不同的元节点，它们各自只能选择一条元链路流入或流出. 这一限制条件将确定每个虚拟节点最终被映射到哪个物理节点上.

但是该算法没有考虑到 SN 中节点之间的差异，笔者认为，如果在映射时对它们区别对待可以提高请求接受率以及 SN 的资源利用率，因此提出了用重要性因子来区分物理节点，计算方法将在 3.2 节详细介绍.

由于混合整数规划的求解是 NP 难问题，所以 Chowdhury 等可通过松弛整数条件将其转化为线性规划模型. 笔者做了进一步改进，具体如下.

1) 变量设定

b_{mn}^{i}是物理链路 l_{mn} 为第 i 条虚拟链路分配的带宽资源. x_{mn} 为二元值，若 $\sum_{i}(b_{mn}^{i}+b_{nm}^{i})>0$ 则为 1；否则为 0，整数条件松弛后为 0～1 之间的小数.

2) 目标函数定义

扩展式(5)，以最小化每个 VNR 的映射开销

$$\min\Bigg\{\sum_{l_{mn}\in E^{S}}\frac{1}{R_{\mathrm{E}}(l_{mn})+\delta}\sum_{i}b_{mn}^{i}+\sum_{s\in N^{S}}\frac{\mathrm{Imp}(s)}{R_{\mathrm{N,C}}(s)+\delta}\sum_{z\in N^{S'}\setminus N^{S}}x_{zs}c(z)+\sum_{s\in N^{S}}\frac{\mathrm{Imp}(s)}{R_{\mathrm{N,H}}(s)+\delta}\sum_{z\in N^{S'}\setminus N^{S}}x_{zs}h(z)\Bigg\} \quad (8)$$

其中 $R_{\mathrm{E}}(l_{mn})$、$R_{\mathrm{N,C}}(s)$ 和 $R_{\mathrm{N,H}}(s)$ 分别为物理链路 l_{mn} 的剩余带宽资源、物理节点 s 剩余的 CPU 和流表资源. 各项分别除以对应的剩余资源值，以保证那些可用资源更多的物理节点和链路更容易被选中，从而达到负载均衡的效果. 极小值 δ 用于防止分母为零. $\mathrm{Imp}(s)$为物理节点 s 的重要性因子.

3) 约束条件

① 资源约束

对 $\forall m,n\in N^{S'}$，$\forall z\in N^{S'}\setminus N^{S}$，$\forall s\in N^{S}$，有

$$\sum_{i}(b_{mn}^{i}+b_{nm}^{i})\leqslant R_{\mathrm{E}}(l_{mn})x_{mn} \quad (9)$$

$$x_{zs}c(z)\leqslant R_{\mathrm{N,C}}(s) \quad (10)$$

$$x_{zs}h(z)\leqslant R_{\mathrm{N,H}}(s) \quad (11)$$

以上 3 个式子包含了链路的带宽资源约束、节点的 CPU 和流表资源约束. 其中式(9)确保物理链路 l_{mn} 所有方向上的流量总和不超过其可用带宽资源.

② 流量约束

$$\sum_{n\in N^{S'}}b_{mn}^{i}-\sum_{n\in N^{S'}}b_{nm}^{i}=0,\ m\in N^{S'}\setminus\{s_i,t_i\} \quad (12)$$

$$\sum_{n\in N^{S'}}b_{s_in}^{i}-\sum_{n\in N^{S'}}b_{ns_i}^{i}=b(e_i^{V}) \quad (13)$$

$$\sum_{n\in N^{S'}}b_{t_in}^{i}-\sum_{n\in N^{S'}}b_{nt_i}^{i}=-b(e_i^{V}) \quad (14)$$

其中：$b(e_i^{V})$表示虚拟网络 G^{V} 的第 i 条虚拟链路请求的带宽资源，它的起点和终点分别被映射到了物理节点 s_i 和 t_i 上. 式(12)～(14)表明，除源节点 s_i 和宿节点 t_i 外，每个节点流入和流出的流量应该相等.

③ 其他约束

$$\sum_{s\in\phi(z)}x_{zs}=1,\ \forall z\in N^{S'}\setminus N^{S} \quad (15)$$

$$\sum_{z\in N^{S'}\setminus N^{S}}x_{zs}\leqslant 1,\ \forall s\in N^{S} \quad (16)$$

式(15)和(16)分别表示确保每个元节点只选择一个物理节点及每个物理节点最多只被一个元节点选中[6].

3　SDN-ScaSVNE 算法

3.1　SDN-ScaSVNE 算法框架

1) 度量物理节点重要性. 计算节点的重要性因子，并将节点划分为重要节点和非重要节点两类.

2) 扩展 VNR. 使用 k 冗余备份策略[2]得到 A-VNR.

3) 使用 SDN-BackupVNE 算法映射 A-VNR.

4) 使用 SDN-ScaREC 算法完成故障恢复.

3.2　物理节点重要性因子

现有针对 SDN 的 VNE 算法并未给出节点重要性计算方法，笔者从资源重要性和网络拓扑重要性两方面进行考虑.

参照文献[1]，对于每个物理节点，资源重要性

的度量需要考虑节点自身资源以及邻居链路带宽和,而在SDN中节点自身资源包括CPU和流表资源. 为了简化计算过程,网络拓扑重要性的度量则仅考虑邻居节点对该节点的贡献. 对 $\forall s \in N^S$,计算方法如下:

1) 计算资源重要性

$$I_{\text{cap}}(s) = [\lambda c(s) + (1-\lambda)h(s)] \sum_{w \in N_{\text{adj}}(s)} b(l_{sw}) \tag{17}$$

其中:$N_{\text{adj}}(s)$为物理节点 s 的邻居节点集;l_{sw}为物理节点 s 和邻居节点 w 的相连链路;λ 为可调参数,用于调节节点自身资源中CPU和流表的重要性比例,本方案取值0.6,即CPU比流表资源更重要.

2) 计算网络拓扑重要性

$$I_{\text{top}}(s) = \sum_{w \in N_{\text{adj}}(s)} \frac{I_{\text{cap}}(w)}{\deg(w)} \tag{18}$$

其中 $\deg(w)$ 表示邻居节点 w 的度.

3) 根据式(17)和(18)得出节点重要性因子

$$\text{Imp}(s) = I_{\text{cap}}(s) + I_{\text{top}}(s) \tag{19}$$

4) 对于每个节点,若满足式(20),则将节点加入重要节点集合 N^S_{Imp},否则加入 N^S_{Nimp}

$$\text{Imp}(s) \geqslant \min_{w \in N^S} \{\text{Imp}(w)\} + \eta\left(\max_{w \in N^S} \{\text{Imp}(w)\} - \min_{w \in N^S} \{\text{Imp}(w)\}\right) \tag{20}$$

其中:η 用于调节节点重要性因子的阈值,超过这个阈值则为重要节点;否则为非重要节点. η 取值0.7.

5) 对节点的重要性因子进行归一化处理

$$\text{Imp}(s) = \frac{\text{Imp}(s)}{\max_{w \in N^S} \{\text{Imp}(w)\}} \tag{21}$$

3.3 映射算法 SDN-BackupVNE

改进了随机型映射算法 R-ViNE[6],使其适用于A-VNR的映射.

1) 构造并求解第2节描述的线性规划模型.

2) 使用R-ViNE中的随机型方式映射虚拟节点.

3) 虚拟节点备份映射. 在虚拟节点的可映射节点集 $\phi(v)$ 中,若存在一个或多个物理节点已为其他虚拟节点提供了备份映射,则从这些节点中选择提供备份资源最多的作为当前虚拟节点的备份映射节点(这种共享策略可以有效降低资源消耗);否则使用随机型方式完成节点的备份映射. 若 $\phi(v)$ 为空,则不备份该虚拟节点,实现了虚拟节点的可伸缩备份.

4) 虚拟链路映射. 若VNR支持路径分裂,使用多商品流(MFC, multi-commodity flow)算法完成链路映射;否则使用 K 最短路径(KSP, K shortest paths)算法完成链路映射.

5) 虚拟链路备份映射. 对存在备份节点的虚拟节点,使用KSP算法为它的每条关联虚拟链路提供备份路径. 如果某些关联链路无可映射的备份路径,则不进行备份,实现了虚拟链路的可伸缩备份.

3.4 故障恢复算法 SDN-ScaREC

不同于文献[2-3]中仅使用备份资源的主动保护策略,笔者通过引入文献[4]中的被动恢复策略,利用备份资源和SN中的剩余资源共同完成故障恢复,改善了主动保护策略恢复资源单一的问题,有效提高了故障恢复成功率. 具体步骤如下.

1) 在物理节点发生故障后,如果受到影响的虚拟节点和链路存在足够的备份资源,则直接将它们迁移至备份资源上,完成故障恢复.

2) 若备份资源不够,则利用SN中的剩余资源,并采取重映射的方式,完成故障恢复.

3) 对于失效虚拟节点,根据式(22)获得可重映射的物理节点集合,这些物理节点满足位置约束且尚未映射该VNR内的其他虚拟节点.

$$\phi'(v) = \left\{ s \in N^S \middle\backslash \left\{ \sum_{v \in N^V} N_M(v) \right\} \middle| \text{Dis}(\text{loc}(v), \text{loc}(s)) \leqslant r^v \right\} \tag{22}$$

其中 $N_M(v)$ 表示被虚拟节点 v 映射的物理节点.

4) 使用上述SDN-BackupVNE算法中的步骤3完成失效虚拟节点的重映射.

5) 为了减少故障恢复的消耗时间,使用复杂度较低的KSP算法重映射失效的虚拟链路.

4 实验仿真

4.1 仿真环境配置

物理SDN拓扑是系数为8的fat-tree类型,包含80个物理节点和256条物理链路. 参照文献[6]设定,物理节点的CPU、流表和链路的带宽资源大小均服从50~100之间的均匀分布. VNR到达规律服从系数为4的泊松分布,仿真持续时间为5万个时间单元,平均每100个时间单元内有4个VNR到达,在5万个时间单元内共计到达2 000个VNR. 每个VNR的虚拟节点个数服从2~10之间的均匀分

布，虚拟节点的连通性为 0.5，虚拟节点的 CPU、流表和链路的带宽资源大小均服从 1 ~ 10 之间的均匀分布.

仿真过程模拟了 300 个物理节点故障，物理节点故障的产生规律服从系数为 0.6 的泊松分布.

4.2　对比方案和性能选取

选取了主动保护策略 LC-SVNE 算法[3] 和被动恢复策略 NB-SVNE 算法[4] 作为对比方案，请求接受率、收益/成本（R/C）比、故障恢复成功率和故障恢复平均耗时作为性能评价指标. 其中，R/C 比值为

$$R/C = \frac{\text{Profit}(T)}{\text{Cost}(T)} \tag{23}$$

4.3　仿真结果和分析

图 2 所示为 SDN-ScaSVNE 和两组对比算法的请求接受率对比. NB-SVNE 仅使用被动恢复策略，不需要扩展虚拟网络，映射单个 VNR 的消耗最低，所以请求接受率最高. LC-SVNE 仅使用主动保护策略，增加了单个 VNR 的映射消耗，请求接受率最低. 而 SDN-ScaSVNE 基于剩余网络资源实现了 VNR 的可伸缩备份，提高了映射成功率.

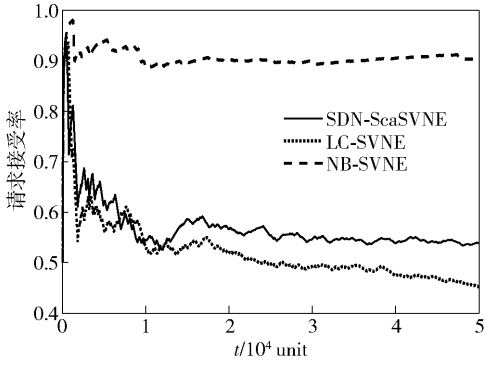

图 2　虚拟网络请求接受率对比

R/C 能够较好地反映算法的总体性能. 图 3 中给出了 3 组算法的 R/C 比值. SDN-ScaSVNE 和 NB-SVNE 的 R/C 比值接近，且明显高于 LC-SVNE. 这是因为 SDN-ScaSVNE 虽然请求接受率低于 NB-SVNE，但是其故障恢复性能更佳. LC-SVNE 因为收益较低，且增加了映射消耗，所以最终的 R/C 比值最低.

图 4 所示为 3 组算法的故障恢复成功率，其中 SDN-ScaSVNE 最高，而 LC-SVNE 最低. 这是因为 SDN-ScaSVNE 算法能够结合已有的备份资源和网络剩余资源共同完成故障恢复，所以相对于仅使用

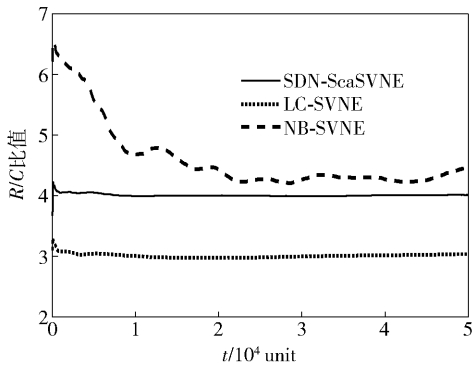

图 3　收益/成本比值对比

备份资源的 LC-SVNE 算法和仅使用网络剩余资源的 NB-SVNE 算法，故障恢复策略更灵活，恢复成功率更高.

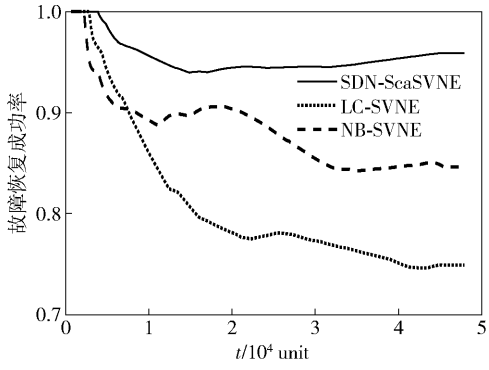

图 4　故障恢复成功率对比

此外，SDN-ScaSVNE、LC-SVNE 和 NB-SVNE 算法单个故障恢复的平均耗时分别为 2.56、2.02 和 3.18 min. 可见，SDN-ScaSVNE 耗时介于仅使用备份资源进行恢复的 LC-SVNE 和仅使用网络剩余资源进行恢复的 NB-SVNE 之间. 这是因为 SDN-ScaSVNE 在备份资源充足时直接使用备份资源进行恢复，备份资源不足时又通过被动恢复策略寻找网络剩余资源进行恢复.

综上，SDN-ScaSVNE 具有最高的故障恢复成功率和 R/C 比值. 但由于需要预留备份资源，所以请求接受率偏低，因此如何确定备份资源的比例，以获得较高的请求接受率还有待进一步研究.

笔者还针对故障物理节点中重要节点的比例不同，对故障恢复成功率和 VNR 请求接受率的影响进行了仿真和分析，如图 5 和图 6 所示.

当故障节点中重要节点比例较低时，SDN-ScaSVNE 的故障恢复成功率较低，而请求接受率较高. 由于 VNR 中重要的虚拟节点往往是所需资源比较

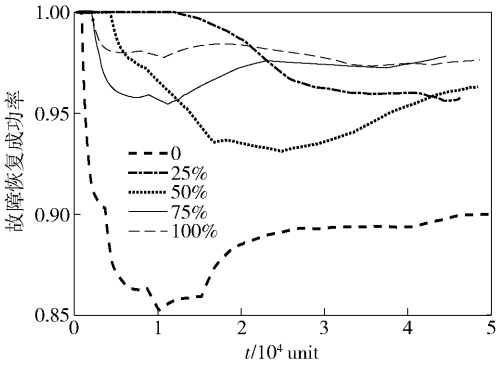

图5 故障节点中重要节点比例对 SDN-ScaSVNE 故障恢复成功率的影响

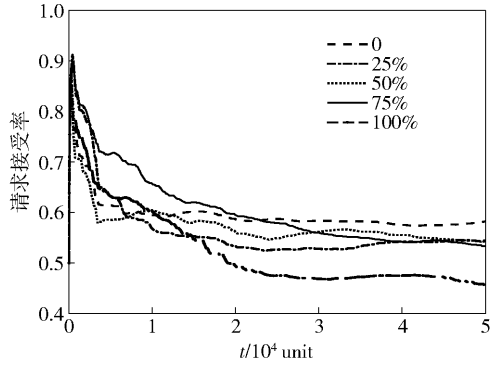

图6 故障节点中重要节点比例对 SDN-ScaSVNE 请求接受率的影响

多或者处在网络关键位置的节点,它们大多会被映射到重要的物理节点上,所以如果重要节点出现故障的比例较低,受影响的虚拟节点也大多是非重要的. 由于本文方案只对虚拟网络中的重要节点提供备份资源,对非重要节点故障只能通过重映射进行故障恢复,导致故障恢复成功率较低. 但失效的虚拟网络因为无法恢复会释放原来占用的物理资源,有利于后续 VNR 的映射,所以请求接受率得到了提升.

随着故障节点中重要节点比例的提高,故障恢复成功率会有很大提升,请求接受率却相应下降. 由上述分析可知,如果产生故障的物理节点大多是重要节点,那么受到影响的虚拟节点也大多是重要的. 笔者提出的方案利用 SN 中的备份资源和可用资源共同完成故障恢复,提高了故障恢复成功率. 但故障节点和成功恢复的虚拟网络会进一步压缩物理网络的可用资源,不利于后续 VNR 的映射,导致请求接受率较低.

上述结果说明,当物理网络中的重要节点发生故障时,SDN-ScaSVNE 能够更加有效地为虚拟网络提供生存性保障,当然由于需要一定的备份资源,会牺牲一定的请求接受率.

5 结束语

结合主动保护策略和基于网络剩余资源的被动保护策略提出了可伸缩备份的 SVNE 算法,改善了现有算法映射僵化和恢复资源单一等问题,能够为虚拟网络提供较好的生存性保障. 后续工作可以从以下 2 个方面进行改进. 首先,VNR 映射收益和故障赔偿金中的参数是人为定义的,需要参考网络运营商的实际数据进行调整;其次,虚拟网络映射和故障恢复阶段的重映射仅采用了静态映射方式,如果在物理网络资源不足的情况下通过动态调整资源分配即动态映射方式可以进一步提高 VNR 请求接受率和故障恢复成功率.

参考文献:

[1] Blenk A, Basta A, Reisslein M, et al. Survey on network virtualization hypervisors for software defined networking [J]. IEEE Communications Surveys and Tutorials, 2016, 18(1): 655-685.

[2] Yu H, Anand V, Qiao C, et al. Cost efficient design of survivable virtual infrastructure to recover from facility node failures [C] // IEEE International Conference on Communications. Kyoto: IEEE, 2011: 1-6.

[3] Hu Q, Wang Y, Cao X. Location-constrained survivable network virtualization[C] // Sarnoff Symposium. Newark: IEEE, 2012: 1-5.

[4] Bo L U, Tao H, Sun X C, et al. Dynamic recovery for survivable virtual network embedding [J]. Journal of China Universities of Posts & Telecommunications, 2014, 21(3): 77-84.

[5] Xiao X C, Zheng X W. A proposal of survivable virtual network embedding algorithm[J]. Journal of High Speed Networks, 2016, 22(3): 241-251.

[6] 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.