动态拓扑下的可重构服务承载网资源迁移方法*

王志明，汪斌强，王保进
（国家数字交换系统工程技术研究中心，郑州 450002）

摘　要：提出了动态网络模型分析承载网资源迁移问题，并引入了迁移效率概念，设计了资源迁移方法。该方法以元请求迁移过程为基础，分别针对拓扑收缩和增长情况提出资源容错迁移和资源均衡迁移两种算法。仿真结果表明，基于迁移效率的资源迁移方法提高了请求接受率和负载均衡度，同时降低了迁移代价。

关键词：可重构服务承载网；动态拓扑；迁移效率；容错；均衡
中图分类号：TP393　　**文献标志码**：A　　**文章编号**：1001-3695(2012)10-3883-05
doi:10.3969/j.issn.1001-3695.2012.10.074

Resource migration scheme for reconfigurable service carrying network under dynamic topology condition

WANG Zhi-ming, WANG Bin-qiang, WANG Bao-jin
(*National Digital Switching System Engineering & Technology Research Center*, *Zhengzhou* 450002, *China*)

Abstract: This paper proposed a dynamic network model to analyze the RSCN resource migration problem, and introduced a concept of migration efficiency to design a resource migration scheme. This scheme included two algorithms based on atomic request migration procedures, which were resource fault-tolerant migration and resource balanced migration respectively focusing on topology contraction and growth. Simulation results show that the migration-efficiency-based resource migration scheme (ME-Mig) increases the request accepted ratio and the network load balance degree, while decreasing the cost of RSCN's migration.

Key words: reconfigurable service carrying network (RSCN); dynamic topology; migration efficiency; fault-tolerance; balanced

0　引言

随着用户需求日趋多样化增长，传统互联网依靠设备升级、带宽扩展以及增加新的协议算法等方式来满足业务需求导致了网络体系结构更加复杂化。其原因在于当前网络是僵化的[1]，升级与扩展是适应新业务的仅有手段。面向服务提供的可重构柔性网络（reconfigurable flexible network based on service providing，FlexNet）[2]以构件化可重构路由交换平台[3]为支撑，通过在同一柔性网络之上构建多个可重构服务承载网（RSCN）的方式来实现多种异构网络的并存与管理，从而为解决网络僵化问题提供一种创新思路。

可重构服务承载网映射技术是柔性网络理论体系的关键内容，该技术主要完成将承载网构建请求映射到柔性网络空闲资源之上的过程。然而，当柔性网络拓扑结构动态变化时，局部资源会出现失效和分配不均衡等问题，从而影响其上承载网服务的提供以及后续构建请求的映射。如何使相应承载网的映射方案适应柔性网络拓扑变化并使资源得到最优化迁移，是现有映射技术所面临的挑战。

目前，国内外相关研究主要围绕虚拟网[4]映射（又称赋值、资源分配）算法。虚拟网映射算法大致分为静态和动态两类[5]方式。前者在虚拟网的活动周期中对映射方案不作任何调整，后者能够根据物理网络和虚拟网的当前状态对映射方案进行资源调整。对于静态方式，Ricci 等人[6]最早提出了 Assign 算法求解实验床中的子网资源分配问题；随后的研究大多将虚拟网映射问题分解为节点映射和链路映射两个步骤，并分别提出各种启发式算法和线性规划模型[7~9]；最近的研究正试图逐步将节点映射与链路映射关联起来，如 Chowdhury 等人[10]提出的 D-ViNE 和 R-ViNE 算法，以及 Jens 等人[11]的子图同构检测 vnmFlib 算法。对于动态方式，He 等人[12]提出了依据虚拟网性能动态调整链路带宽分配的 DaVinci 算法；Rahman 等人[13]针对物理拓扑故障提出了一种保证虚拟网生存性的映射算法；Cai 等人[14]则针对物理拓扑变化的一般情况，提出一种基于代价最小原则的节点迁移与重映射算法。

上述研究的不足之处在于：大多基于静态的网络模型对拓扑变化情况未能有效地描述与分析；静态网络模型下的迁移策略也相对单一，只考虑了最短路径、负载均衡或者迁移代价等，缺乏更为合理的迁移策略；另外，对于拓扑增长情况下的新增资源利用，当前研究主要是等待新构建请求到达之后再利用之，而缺乏一种有效的主动利用方式。

针对不足，本文给出一种描述可重构服务承载网拓扑变化的动态网络模型，并在此基础上提出基于迁移效率的资源迁移方法（ME-Mig）。该方法包括基于迁移效率的元请求迁移过

收稿日期：2012-02-14；**修回日期**：2012-03-22　　**基金项目**：国家“863”计划基金资助项目（2008AA01A323，2009AA01A334）；国家“973”计划资助项目（2012CB315900）

作者简介：王志明（1986-），男，辽宁大连人，博士研究生，主要研究方向为网络虚拟化、下一代互联网技术（wangzm05@gmail.com）；汪斌强（1963-），男，教授，博导，主要研究方向为宽带信息网络；王保进（1974-），男，讲师，主要研究方向为网络管控、嵌入式系统.

程、针对拓扑收缩情况的资源容错迁移算法,以及针对拓扑增长情况的资源均衡迁移算法。元请求迁移过程以迁移效率最大为原则,综合考虑负载均衡和迁移代价,实现单个节点/链路请求的最优化迁移。在此基础上提出的资源容错迁移和资源均衡迁移则有效改善了柔性网络拓扑变化时的资源失效和分配不均衡等问题,使承载网映射方案更好地适应新拓扑的资源布局。经过仿真验证,本文提出的 ME-Mig 方法能够以较少的迁移代价使网络资源得到更高效的利用。

1 模型分析

1.1 网络模型

FlexNet 的拓扑结构是动态变化的,所以 RSCN 映射问题需要建立如下的动态网络模型:

a)可重构柔性网络。FlexNet 在拓扑变化前后可以用无向图 $G^s(t^-)=(N_1^s,L_1^s,A_1^N,A_1^L)$ 和 $G^s(t)=(N_2^s,L_2^s,A_2^N,A_2^L)$ 表示。其中:N_1^s 和 N_2^s 分别为拓扑变化前后的节点集合;L_1^s 和 L_2^s 为链路集合;A_1^N 和 A_2^N 为节点资源属性,如 CPU 资源、存储资源等;A_1^L 和 A_2^L 为链路资源属性,如带宽、延迟等。本文只考虑节点的 CPU 资源和链路的带宽资源,所以资源属性体现为一个数值,如图 1(b)中所示。

b)可重构服务承载网构建请求。无向图 $G_i^v=(N_i^v,L_i^v,C_i^N,C_i^L)$ 表示第 i 个到达的 RSCN 构建请求。其中:N_i^v 和 L_i^v 分别为该构建请求的节点集合和链路集合;C_i^N 为节点的资源属性约束;C_i^L 为链路的资源属性约束。如图 1(a)所示,构建请求 1 是为了建立一个连通三个物理节点的 RSCN,且需要满足如下约束:三个物理节点的资源总量均不小于 20,三条物理路径的资源总量分别不小于 15、20 和 25。

c)可重构服务承载网映射。t 时刻第 i 个 RSCN 的映射方案 $f_{i,t}$ 可以描述为从 G_i^v 到 $G^s(t)=(N^s,L^s,A^N,A^L)$ 子集 $G_i^{\text{sub}}(t)$ 的函数关系,且满足资源属性约束 C_i^N 和 C_i^L,表示为

$$f_{i,t}:G_i^v\mapsto G_i^{\text{sub}}(t) \tag{1}$$

其中:$G_i^{\text{sub}}(t)=(N_i^{\text{sub}},P_i^{\text{sub}},R_i^N,R_i^L)$;$N_i^{\text{sub}}\subseteq N^s$;$P_i^{\text{sub}}\subseteq P^s$;$P_i^{\text{sub}}$ 和 P^s 分别表示 $G_i^{\text{sub}}(t)$ 和 $G^s(t)$ 中所有路径的集合;R_i^N 和 R_i^L 分别表示该请求所获得的节点资源和链路资源。

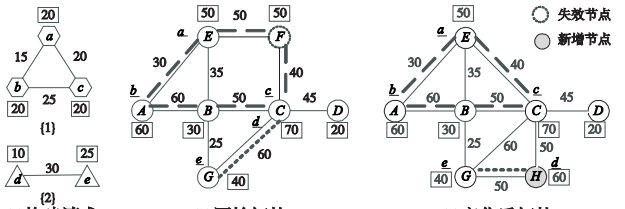
式(1)的映射形式可分解为节点映射和链路映射两部分。

节点映射: $$f_{i,t}^N:(N_i^v,C_i^N)\mapsto(N_i^{\text{sub}},R_i^N) \tag{2}$$

链路映射: $$f_{i,t}^L:(L_i^v,C_i^L)\mapsto(P_i^{\text{sub}},R_i^L) \tag{3}$$

为表述方便,后续公式省略式(2)(3)中的 C_i^N、R_i^N 和 C_i^L、R_i^L。图 1(a)中的构建请求映射方案如图 1(b)所示。构建请求 1 的映射方案为:节点请求$(a,20)$、$(b,20)$和$(c,20)$分别映射到物理节点$(E,20)$、$(A,20)$和$(C,20)$之上,链路请求$(ab,15)$、$(bc,25)$和$(ac,20)$分别映射到物理路径$(\{EA\},15)$、$(\{AB,BC\},25)$和$(\{EF,FC\},20)$之上。

RSCN 构建请求是在线到达的,可以认为满足 M/M/1 排队模型,且 FlexNet 的物理拓扑也在动态变化中,所以 FlexNet 的映射方案和负载状态均随时间变化。图 1 中,FlexNet 的拓扑结构由(b)演变为(c),其间节点 F 和链路 EF、FC 被移除,节点 H 和链路 CH、GH 被增加;原有的映射方案也随之改变,链路请求 ac 由$\{EF,FC\}$迁移到$\{EC\}$,节点请求 d 由 C 迁移到 H,链路请求 de 也由$\{CG\}$迁移到$\{HG\}$。为进一步刻画 RSCN 映射问题的动态性,提出如下概念。



(a)构建请求 (b)原始拓扑 (c)变化后拓扑

图1 拓扑变化情况下的可重构服务承载网构建与迁移

定义 1 映射族。t 时刻 FlexNet 中所有 RSCN 的映射方案集合 F_t。$|F_t|$等于 t 时刻 FlexNet 中现存的 RSCN 个数,且 F_t 满足如下的递推关系:

$$F_t=\begin{cases}\varnothing & t=0\\ F_{t^-}\cup\{f_{k,t}\} & t=a_k\\ F_{t^-}\setminus\{f_{k,t}\} & t=d_k\\ (F_{t^-}\setminus\{f_{k,t^-}\})\cup\{f_{k,t}\} & t=r_k\\ F_{t^-} & t=\text{else}\end{cases} \tag{4}$$

其中:a_k 为第 k 个 RSCN 构建成功的时刻;d_k 为第 k 个 RSCN 注销的时刻;r_k 为第 k 个 RSCN 发生迁移的时刻;k 为正整数;$t^-<t$,且$[t^-,t)$时间段内未发生 RSCN 的构建、注销及迁移。

定义 2 节点压力。即节点分配给各 RSCN 的资源之和占该节点资源总量的百分比。t 时刻物理节点 u 的压力可表示为

$$W_N(t,u)=\frac{\sum_{i\in s}|R_i^N(u)|}{|A^N(u)|} \tag{5}$$

其中:$s=\{k|\exists\hat{u}\in N_k^v,f_{k,t}^N(\hat{u})=u\}$,表示物理节点 u 所承载的 RSCN 序列集合。

定义 3 链路压力。即链路分配给各 RSCN 的资源之和占该链路资源总量的百分比。t 时刻物理链路 e 的压力可表示为

$$W_L(t,e)=\frac{\sum_{i\in s}|R_i^L(e)|}{|A^L(e)|} \tag{6}$$

其中:$s=\{k|\exists\hat{e}\in L_k^v,f_{k,t}^L(\hat{e})=p,e\in p\}$,表示物理链路 e 所承载的 RSCN 序列集合。

由式(5)(6)可知,FlexNet 的节点/链路压力随时间 t 不断变化,特别是在 RSCN 构建、注销和迁移的时刻,涉及的节点/链路会因为承载的 RSCN 发生变化而出现节点/链路压力的增减。

1.2 问题描述

对于 RSCN 资源迁移问题,既要考虑高效利用新增的节点/链路资源,又要避免因局部节点/链路异常所带来的 RSCN 服务质量下降,同时尽量减少迁移所产生的代价。所以在解决资源迁移问题时,需综合考虑负载均衡原则和代价最小原则。

a)负载均衡体现在 FlexNet 节点/链路压力的差异幅度。相关定义如下:

定义 4 节点压力振幅。即各物理节点压力与平均节点压力的绝对值均差。由式(5)可得到 t 时刻 FlexNet 的节点压力振幅为

$$NA(t)=\Big(\sum_{u\in N^s}|W_N(t,u)-\overline{W}_N(t)|\Big)/|N^s| \tag{7}$$

定义 5 链路压力振幅。即各物理链路压力与平均链路压力的绝对值均差。由式(6)可得到 t 时刻 FlexNet 的链路压力振幅为

$$LA(t)=\Big(\sum_{e\in L^s}|W_L(t,e)-\overline{W}_L(t)|\Big)/|L^s| \tag{8}$$

定义 6 负载均衡度。FlexNet 的负载均衡度是节点、链路

压力振幅的线性组合。结合式(7)(8)得到 t 时刻 FlexNet 的负载均衡度 $\pi(t)$,$\pi(t)$越大,负载均衡程度越高。

$$\pi(t)=\alpha\times(1-NA(t))+\beta\times(1-LA(t)) \tag{9}$$

其中:$\alpha,\beta\in(0,1)$为节点和链路压力振幅的权重,且有 $\alpha+\beta=1$ 成立,取值依具体情况。

b)迁移代价体现在 RSCN 迁移过程中涉及的节点/链路变动。节点/链路变动为整个网络带来计算消耗和服务切换消耗,而服务切换消耗又包括节点切换消耗和链路切换消耗。计算消耗是指在迁移过程中运行相关算法所产生的消耗;节点切换是指将一个物理节点的服务处理能力切换到另一个物理节点上,其间伴随节点相关构件的注销与加载;链路切换是指重新配置一条新的物理链路,其间伴随节点的路由配置更改。所以节点切换消耗与发生变动的节点数呈正比,链路切换消耗与新配置的链路数呈正比。

定义 7　迁移代价。设 w_{comp} 为计算消耗,w_n 为单个节点切换消耗,w_l 为单条链路切换消耗,n_{switch}为发生迁移的节点个数,l_{switch}为发生迁移的链路个数。在(t,T)时间段内,拓扑结构由 $G^s(t)$演变为 $G^s(T)$,则每个 RSCN 迁移过程$f_{i,t}\to f_{i,T}$的迁移代价为

$$C(f_{i,T},f_{i,t},G^s(T),G^s(t))=w_{\text{comp}}+w_n\times n_{\text{switch}}+w_l\times l_{\text{switch}} \tag{10}$$

其中:$n_{\text{switch}}=|\{u\mid f^N_{i,t}(u)\neq f^N_{i,T}(u),u\in N^v_i\}|$,$l_{\text{switch}}=\sum_{f^L_{i,t}(e)\neq f^L_{i,T}(e),e\in L^v_i}|f^L_{i,T}(e)-f^L_{i,t}(e)|$。由式(10)进一步推导可知,FlexNet 的映射族从 F_t 变化为 F_T 的迁移代价为

$$C(F_T,F_t,G^s(T),G^s(t))=\sum_{f_{i,T}\neq f_{i,t}}C(f_{i,T},f_{i,t},G^s(T),G^s(t)) \tag{11}$$

其中:$f_{i,T}\in F_T$,$f_{i,t}\in F_t$,且$|F_t|=|F_T|$。

在上述定义的基础上,本文提出迁移效率的概念来刻画 RSCN 资源迁移问题的目标函数。

定义 8　迁移效率。是单位迁移代价所带来的负载均衡度提高,即负载均衡度增量与迁移代价的比值。对于(t,T)时间内的 RSCN 资源迁移过程,其迁移效率可以表示为$\dfrac{\pi(T)-\pi(t)+\varepsilon}{C(F_T,F_t,G^s(T),G^s(t))}$,其中 ε 为极小的整数,以便在负载均衡度增量为 0 时退化为迁移代价最小原则。

综上所述,RSCN 资源迁移问题就是寻找使迁移效率最大的新映射族 $F(T)$,即

$$\max\frac{\pi(T)-\pi(t)+\varepsilon}{C(F_T,F_t,G^s(T),G^s(t))},\ \text{s.t.} \tag{12}$$

$$C^N_i(u)\leqslant R^N_i(v),\forall u\in N^v_i,v\in N^{\text{sub}}_i,\text{if}\quad f^N_{i,T}(u)=v \tag{13}$$

$$C^L_i(e)\leqslant R^L_i(e'),\forall e\in L^v_i,e'\in P^{\text{sub}}_i,\text{if}\quad e'\in f^L_{i,T}(e) \tag{14}$$

如果设 $x_{i,m,n}\in\{0,1\}$表示第 i 个 RSCN 请求是否将第 m 个节点请求映射到第 n 个物理节点,且 $1\leqslant i\leqslant|F_T|$,$1\leqslant m\leqslant|N^v_i|$,$1\leqslant n\leqslant|N^s(T)|$,那么上述问题可以进一步抽象为 0-1 整数规划问题,属于 NP-hard,所以下文采用启发式算法求解。

2　基于迁移效率的资源迁移

2.1　元请求迁移过程

RSCN 映射方案的迁移实际上是对部分节点/链路请求分别进行迁移,如果将单个节点/链路请求称之为元请求(atomic request),那么 RSCN 资源迁移问题可以分解为众多元请求的迁移问题。所以,根据贪心策略,资源迁移问题的最优化求解可以近似为使每个元请求的迁移效率最大。下面将分别给出链路和节点请求迁移的具体过程。

链路请求迁移的目的在于搜索一条迁移效率最高的替代路径,采用 k-shortest paths 算法[15]进行近似求解,一方面是因为其复杂度为多项式级别,另一方面是为了尽量缩短最优迁移路径的长度,因为过长的路径会带来高延迟等问题。其过程如下:

procedure LinkMig($G^s(T)$,G^v_i,f_{old},e):
1 $f_{\text{new}}\leftarrow f_{\text{old}}$;
2 对于待迁移的链路请求 e 所对应物理路径端点 u 和 v,在 $G^s(T)$中使用 k-shortest paths 算法得到 u 与 v 之间的 k 条最短路径 $P=\{p_1,p_2,\cdots,p_k\}$;
3 选取集合 P 中满足约束条件式(14),且使式(12)最大的路径 p_{best};
4 if p_{best}不存在 then
5　return NULL;
6 else
7　更新f_{new},使$f^L_{\text{new}}(e)=p_{\text{best}}$;
8 end if
9 return f_{new};

节点请求迁移涉及到与之相邻的链路请求,所以分为两个步骤:a)将节点请求迁移到资源空闲度(定义 9)最大的物理节点;b)完成与节点请求相邻的链路请求迁移,调用的是 LinkMig 过程。资源空闲度越大,节点请求以及相邻的链路请求越容易得到满足,剩余资源也越多,从而既提高了节点请求迁移的成功率,又能尽量避免迁移之后的节点压力过载。

定义 9　节点资源空闲度。即节点剩余资源与所有相邻链路剩余资源和的乘积。对于物理节点 u,其资源空闲度为 idle(u),相邻链路集合为 neigh(u),则有

$$\text{idle}(u)=A^N(u)\times(1-W_N(t,u))\times\sum_{e\in\text{neigh}(u)}A^L(e)\times(1-W_L(t,e)) \tag{15}$$

下面是节点请求迁移(NodeMig)的具体过程:

procedure NodeMig($G^s(T)$,G^v_i,f_{old},v):
1 $f_{\text{new}}\leftarrow f_{\text{old}}$;
2 选取 $G^s(T)$中除节点$f_{\text{old}}(v)$外,满足约束条件式(13),且使式(15)最大的节点 u_{best};
3 if u_{best}不存在 then
4　return NULL;
5 else
6　更新f_{new}, 使$f^N_{\text{new}}(v)=u_{\text{best}}$;
7 end if
8 for each e 为与 v 相连的链路请求 do
9　$f_{\text{new}}\leftarrow$ LinkMig ($G^s(T)$,G^v_i,f_{new},e); //相邻链路请求迁移
10　if f_{new}为 NULL then
11　　return NULL;
12　end if
13 end for
14 return f_{new};

由于 k-shortest paths 算法的时间复杂度为 $O(N\log N+kN+M)$[15],其中 N 为拓扑 $G^s(T)$的节点个数,M 为链路个数,所以 LinkMig()过程的时间复杂度同样为 $O(N\log N+kN+M)$。若设 m 为所有 RSCN 请求中节点个数的最大值,则 NodeMig()过程在最坏情况下的时间复杂度为 $O(m(N\log N+kN+M))$。下面以元请求迁移过程为基础,进一步给出资源容错迁移和资源均衡迁移算法。

2.2　资源容错迁移

当 FlexNet 拓扑收缩时,部分物理节点/链路将失效,从而影响其上承载的 RSCN 正常服务的提供。当物理节点 u 失效时,所有与 u 相关联的元请求需要进行迁移,其中包括节点 u 所承载的节点请求和以节点 u 为中间节点的链路请求;当物理

链路 e' 失效时，所有与 e' 相关联的链路请求需要进行迁移。以下为资源容错迁移算法的具体过程：

算法 1 资源容错迁移（resource fault-tolerant migration，RFT_MIG）

输入：$G^s(t)$，$G^s(T)$，F_t，G_i^v，$i=1,2,\cdots$。

输出：F_T。

```
1 初始化 F_T = F_t;
2 if u ∈ N^s(t) 失效 then //节点资源容错迁移
3 得到物理节点 u 所关联的节点映射 S = {(f_{i,t}, v) | f^N_{i,t}(v) = u, v ∈ N_i^v, f_{i,t} ∈ F_t};
4 for each (f_{i,t}, v) ∈ S do
5   f_new ← NodeMig(G^s(T), G_i^v, f_{i,t}, v);
6   若节点迁移成功，更新 F_T，令 F_T ← (F_T \ {f_{i,t}}) ∪ {f_new};
7 end for
8 for each f_{i,t} ∈ F_t do
9   得到 G_i^v 中受物理节点 u 影响的链路请求 E = {e | f^L_{i,t}(e) = p, e ∈ L_i^v, u 为 p 的中间节点};
10  f_new ← f_{i,t};
11  for each e ∈ E do
12    f_new ← LinkMig(G^s(T), G_i^v, f_new, e);
13  end for
14  F_T ← (F_T \ {f_{i,t}}) ∪ {f_new};
15 end for
16 else if e' ∈ L^s(t) 失效 then //链路资源容错迁移
17 for each f_{i,t} ∈ F_t do
18  得到 G_i^v 中受物理链路 e' 影响的链路请求 E = {e | f^L_{i,t}(e) = p, e ∈ L_i^v, e' ∈ p};
19  f_new ← f_{i,t};
20  for eache ∈ E do
21    f_new ← LinkMig(G^s(T), G_i^v, f_new, e);
22  end for
23  F_T ← (F_T \ {f_{i,t}}) ∪ {f_new};
24 end for
25 end if
26 return F_T;
```

由 NodeMig 和 LinkMig 过程的时间复杂度可知，RFT_MIG 算法在解决节点失效情况时，最坏情况时间复杂度为 $O(n(m+l)(N\log N+kN+M))$；在解决链路失效情况时，最坏情况时间复杂度为 $O(nl(N\log N+kN+M))$，其中 $n=|F_t|$，m 和 l 分别为所有 RSCN 请求中节点和链路个数的最大值。

2.3 资源均衡迁移

当 FlexNet 拓扑增长时，新的节点/链路资源将被增加，而部分节点/链路压力过载。针对这一情况，提出资源均衡迁移算法，其基本思路是：对节点/链路压力状况进行多次检测，每次选出压力最大的过载节点/链路；针对该节点/链路，仅将其上资源约束最大的一个元请求进行局部迁移，从而以最少的迁移次数来最大限度地缓解过载节点/链路的资源压力。

压力超过阈值的节点/链路称为过载节点/链路，其中节点和链路压力阈值分别设为 δ_N 和 δ_L。另外，由于每次迁移之后压力分布会出现变化，调度位置需要重新检测。为减少迁移代价和 RSCN 抖动，设压力过载检测的循环次数不超过 MAX_SCHED_WINDOW（正整数）。资源均衡迁移的具体过程如下：

算法 2 资源均衡迁移（resource balanced migration，RBM_MIG）

输入：$G^s(t)$，$G^s(T)$，F_t，G_i^v，$i=1,2,\cdots$。

输出：F_T。

```
1 F_T ← F_t;
2 for i = 1,2,…,MAX_SCHEDULE_WINDOW do
3   得到压力过载节点集合 M_N = {u | W_N(t,u) > δ_N, u ∈ N^s(t)};
4   if M_N = ∅ then break; end if
5   得到 M_N 中压力最大的节点 v，及其所关联的节点映射 S = {(f_{i,T}, w) | f^N_{i,T}(w) = v, w ∈ N_i^v};
6   搜索 S 中资源约束最大的节点请求 w_k 和映射关系 (f_{k,T}, w_k);
7   f_new ← NR_MIG(G^s(T), G_i^v, f_{k,T}, w_k);
8   F_T ← (F_T \ {f_{k,T}}) ∪ f_new;
9 end for
10 for i = 1,2,…,MAX_SCHEDULE_WINDOW do
11   得到压力过载链路集合 M_L = {e | W_L(t,e) > δ_L, e ∈ L^s(t)};
12   if M_L = ∅ then break; end if
13   得到 M_L 中压力最大的链路 e'，及其所关联的链路映射 E = {(f_{i,T}, ê) | f^L_{i,T}(ê) = p, e' ∈ p, ê ∈ L_i^v};
14   搜索 E 中资源约束最大的链路请求 ê_k 和映射关系 (f_{k,T}, ê_k);
15   f_new ← LR_MIG(G^s(T), G_i^v, f_{k,T}, ê_k);
16   F_T ← (F_T \ {f_{k,T}}) ∪ f_new;
17 end for
18 return F_T;
```

上述算法在拓扑增长时分别对节点压力和链路压力进行主动的资源均衡迁移，将压力过载的节点/链路所承载的元请求迁移到空闲资源之上。由 NodeMig 和 LinkMig 过程的特点可知，迁移目的位置会优先选择新增节点/链路，从而充分利用新增资源来缓解局部压力过载的情况。因为循环次数为正整数，所以 RBM_MIG 算法的时间复杂度级别与 NodeMig 过程相同。

3 实验评估

3.1 实验设置

实验环境为 Intel® Core™ i7 CPU 2.67 GHz，RAM 2 GB 的 PC 上，通过 C++ 编程模拟 FlexNet 的拓扑变化以及 RSCN 的构建与迁移。使用 GT-ITM[16] 工具生成具有 100 个节点和 570 条边的 FlexNet 初始拓扑，以及 2 000 个 RSCN 请求。FlexNet 的节点和链路资源取值在[50，100]内均匀分布，节点连接概率为 0.5，在整个仿真过程中拓扑结构会随机改变，共出现 50 次节点增加事件、30 次节点失效事件、50 次链路增加事件以及 50 次链路失效事件。RSCN 请求的节点个数在[2，10]内均匀分布，节点和链路资源请求在[0，30]内均匀分布，节点连接概率为 0.5。请求的到达过程服从强度为平均 100 个单位时间到达 20 个请求的泊松过程，每个 RSCN 的生命周期服从 μ 为 1 000 个时间单位的指数分布。此外，对于式(9)负载均衡度计算，取 $\alpha=\beta=0.5$；对于式(10)的迁移代价计算，令 w_{comp} 为 1.0，w_n 为 2.0，w_l 为 1.5；对于节点/链路压力检测，令 $\delta_N=\delta_L=0.9$，MAX_SCHED_WINDOW = 3；对于 LR_MIG 算法，令 $k=10$。

在上述实验设置下对资源迁移方法 ME-Mig 的请求接受率、负载均衡度和迁移代价三项指标进行分析验证。其中，请求接受率为构建成功的请求占所有请求的百分比，即 $R_{accept}=R_{success}/R_{total}$，负载均衡度的计算如式(9)所示，迁移代价的计算如式(10)所示。

3.2 性能分析

参与比较的资源迁移算法包括无任何迁移处理的基本承载网映射算法 No-Mig[8]、采用负载均衡策略的迁移算法 Bal-Mig、采用迁移代价最小策略的迁移算法 Cost-Mig[14]，以及本文提出的基于迁移效率的资源迁移方法 ME-Mig。

根据实验设置，首先得到不同算法下的请求接受率曲线，如图 2 所示。由实验结果可知，当 FlexNet 拓扑静态（No-Topo-Change）时，请求接受率最高，为 52.69%；而当拓扑动态变化时，请求接受率因迁移策略的不同而受到不同程度的影响。其

中,在不进行任何迁移处理(No-Mig)时,请求接受率下降最为明显,为 32.03%;采用 ME-Mig 方法可以使请求接受率达到 51.20%,接近拓扑静态时的数值;采用负载均衡策略的 Bal-Mig 算法次之,达到了 47.62%;而采用迁移代价最小策略的 Cost-Mig 算法请求接受率相对较小,仅为 42.75%。请求接受率的大小主要与整个网络的负载均衡度和剩余资源量有关,ME-Mig 和 Bal-Mig 算法均考虑了负载均衡因素,所以请求接受率较高,且 ME-Mig 方法同时考虑了迁移代价,避免迁移方案占用更多的网络资源,所以优于 Bal-Mig 算法;而 Cost-Mig 算法忽略了负载均衡原则,所以请求接受率处于较低水平,仅优于 No-Mig 算法。

随着 RSCN 请求的映射成功,网络负载均衡度从 1.0 逐渐降低到一定数值,直到仿真的后半阶段,RSCN 请求开始逐渐离开,使得负载均衡度又恢复到 1.0。为了分析正常状态下的网络负载均衡度,本文着重观测仿真的前半阶段(0 ~ 4 500 时间单位),如图 3 所示。由实验结果可知,ME-Mig 方法产生的负载均衡度仅次于 Bal-Mig 算法;而 No-Mig 算法随着 RSCN 请求的不断到达与离开,网络中出现资源碎片,从而负载均衡度急剧下降;Cost-Mig 算法由于不考虑负载均衡因素,所以负载均衡度始终较低。

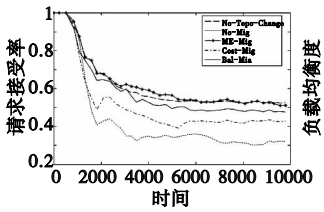


图2　可重构服务承载网构建请求接受率

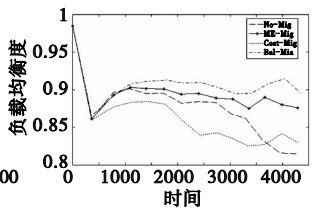


图3　负载均衡度随承载网构建与迁移的变化曲线

各算法得到的总迁移代价变化曲线如图 4 所示。总迁移代价取决于受影响的 RSCN 元请求数和单个元请求迁移所产生的迁移代价。由于 No-Mig 算法不进行迁移,迁移代价为 0;ME-Mig 方法考虑了迁移代价,并且通过负载均衡策略获得了较高的网络负载均衡度,从而降低了平均受影响的 RSCN 元请求数,所以总迁移代价最小;Cost-Mig 算法只考虑了迁移代价,未关注网络负载均衡,所以总迁移代价逐渐高于 ME-Mig 方法;Bal-Mig 算法只考虑网络负载均衡,未关注元请求的迁移代价,所以总迁移代价同样高于 ME-Mig 方法。

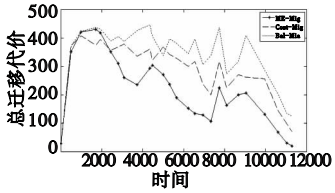


图4　总迁移代价随时间的变化曲线

实验结果表明,基于迁移效率的 RSCN 资源迁移方法 ME-Mig 具有较高的请求接受率、负载均衡度和较低的迁移代价,兼顾了负载均衡和总迁移代价最小两方面的优势,即以最小的迁移代价换来对全网资源的最高效利用。

4　结束语

本文针对可重构柔性网络拓扑动态变化下的资源迁移问题进行了论述,给出一种动态网络模型分析承载网资源迁移问题,并在此基础上综合考虑负载均衡和迁移代价,提出了迁移效率最大原则;然后以该原则为指导,提出元请求迁移过程以及资源容错迁移和资源均衡迁移算法;通过仿真验证,资源迁移方法 ME-Mig 在请求接受率、负载均衡度和迁移代价上具有明显优势,高效利用了网络资源。下一步研究将针对支持路径切分方式的网络进行算法的扩展与优化。

参考文献:

[1] TURNER J S, TAYLOR D E. Diversifying the Internet[C]//Proc of Global Telecommunications Conference. 2005:755-760.

[2] HU Yu-xiang, LAN Ju-long, WU Jiang-xing. Providing personalized converged services based on flexible network reconfiguration[J]. Science China Information Sciences, 2011, 54(2):334-347.

[3] 李玉峰,邱菡,兰巨龙. 可重构路由器研究的现状与展望[J]. 中国工程科学,2008,10(7):82-89,95.

[4] ANDERSON T, PETERSON L, SHENKER S, *et al*. Overcoming the Internet impasse through virtualization[J]. Computer, 2005, 38(4): 34-41.

[5] HAIDER A, POTTER R, NAKAO A. Challenges in resource allocation in network virtualization[C]//Proc of the 20th ITC Specialist Seminar on Network Virtualization. 2009.

[6] RICCI R, ALFELD C, LEPREAU J. A solver for the network testbed mapping problem[J]. ACM Computer Communication Review, 2003, 33(2):65-81.

[7] YU Min-lan, YI Y, REXFORD J, *et al*. Rethinking virtual network embedding: substrate support for path splitting and migration[J]. ACM SIGCOMM Computer Communications Review, 2008, 38(2):17-29.

[8] ZHU Yong, AMMAR M H. Algorithms for assigning substrate network resources to virtual network components[C]//Proc of the 25th IEEE International Conference on Computer Communications. 2006:1-12.

[9] LU Jing, TURNER J. Efficient mapping of virtual networks onto a shared substrate, WUCSE-2006-35[R]. St Louis: Washington University, 2006.

[10] CHOWDHURY N M M K, RAHMAN M R, BOUTABA R. Virtual network embedding with coordinated node and link mapping[C]// Proc of the 28th IEEE INFOCOM. 2009:19-25.

[11] JENS L, HOLGER K. A virtual network mapping algorithm based on subgraph isomorphism detection[C]//Proc of the 1st ACM Workshop on Virtualized Infrastructure Systems and Architectures. New York: ACM Press, 2009:81-88.

[12] HE Jia-yue, SHENG Rui-zhang, LI Ying, *et al*. DaVinci: dynamically adaptive virtual networks for a customized Internet[C]//Proc of ACM CoNEXT Conference. New York: ACM Press, 2008.

[13] RAHMAN M R, AIB I, BOUTABA R. Survivable virtual network embedding[C]//Proc of the 9th IFIP TC 6 International Conference on Networking. Berlin: Spring-Verlag, 2010:40-52.

[14] CAI Zhi-ping, LIU Fang, XIAO Nong, *et al*. Virtual network embedding for evolving networks[C]//Proc of Global Telecommunications Conference. 2010.

[15] EPPSTEIN D. Finding the *k* shortest paths[C]//Proc of the 35th Annual Symposium on Foundations of Computer Science. 1994: 154-165.

[16] ZEGURA E W, CALVERT K L, BHATTACHARJEE S. How to model an internetwork[C]//Proc of the 15th IEEE INFOCOM. Washington DC: IEEE Computer Society, 1996:594-602.