一种基于MPLS流量工程的动态路由算法

王 红，李丽丹，张丽平

（辽宁工程技术大学 电子与信息工程学院，辽宁 葫芦岛 125105）

摘 要：MPLS流量工程的问题最终可以归结为数据流传输的路径确定问题，即显式路径的确立问题。通过对XUE算法的分析，提出了一种新的基于链路和路径的动态路由算法—LPR。依据网络链路平均利用率的取值范围对网络进行裁剪，在选路由时优先选择轻度占用的链路，避开重度占用的链路；从路径的角度出发，计算每条路径中的各链路带宽利用率相对于网络中链路带宽利用率均值的方差。用C++语言完成了该算法的实现，同时验证了该算法较SPF算法及XUE算法的有效性。

关键词：MPLS流量工程；动态路由；链路路径；SPF；XUE

中图分类号:TP393 **文献标识码**：A

A dynamic routing algorithm based on MPLS traffic engineering

WANG Hong, LI Li Dan, ZHANG Li Ping

(School of Electronic and Information Engineering,Liaoning Technical University,Huludao 125105,China)

Abstract: The MPLS traffic engineering question finally may sum up as the data stream transmission path determination question, namely the explicit path establishment question.A new dynamic routing algorithm LPR based on link and path is proposed. Cutting out in view of the network link average using factor value scope to the network, and choosing the link by first which takes mildly, avoiding link which the specific weight takes. According to the point of path, calculating the variance of various links band width using factor in each path relative to the link band width using factor average value in the network. Using the C++ language to complete this algorithm realization, simultaneously it confirmes this algorithm is validity to compare the SPF algorithm and the XUE algorithm.

Key words: MPLS traffic engineering; dynamic routing; link path; SPF; XUE

1 XUE算法

XUE[1]算法即基于带宽和跳数的流量工程动态路由选择算法，是一种最为典型的基于约束路由选择算法（CSPF）[2]。XUE算法采用利用率阀值激活方式，与现有路由算法兼容，以带宽和跳数作为约束，目标函数是使跳数最少。适时激活该算法，根据路由结果建立一条或多条显式路径，将部分流量转移到这些路径上，其时间复杂度与Dijkstra算法相当，是一种有效可行的动态路由算法。但是此算法仅考虑了网络带宽的当前利用状况，虽然在一定程度上避免了链路瓶颈的发生[3]，却忽略了网络资源的均衡分布[4]。由此本文针对XUE算法未考虑的问题给出解决方案，提出了一种新的动态路由算法–LPR。

2 LPR算法实现

2.1 算法描述

该算法从链路和路径两方面考虑。链路的带宽利用率反映链路的负载情况，所有链路的负载状况都可以反映整个网络的流量均衡程度[5]。显然，当网络中所有链路的负载都较轻时，网络不会出现拥塞，所有用户的QoS都能得到保证。在本算法中定义两个常量：j，k （0<j<k<1），根据网络中满足带宽要求的链路带宽利用率平均值的取值情况划分到3个区域中①（0,j],②(j,k],③(k,1]， 若网络中各链路带宽占用率的平均值在区域①，则将网络中所有链路占用率大于j的链路裁剪掉，生成新的网络拓扑v′，若网络中各链路带宽占用率的平均值属于区域②，则将网络中所有链路占用率大于k × u(u的取值范围视k值及具体的网络而定，一般在1的偏右)的链路裁剪掉，生成新的网络拓扑v；若网络中各链路带宽占用的平均值在区域③，则不进行任何的裁剪工作，网络中的拓扑图仍然保持原来的v。这样做的目的是在选路由时优先选择轻度占用的链路，避开重度占用的链路，在加

快算法收敛速度的同时提高了全网的资源利用率；从路径的角度出发，提出了路径均衡度的概念，即针对第 K 最短路径（以跳数为度量）计算出来的 i 条路径，分别计算每条路径中的各链路带宽利用率相对于网络中各链路带宽利用率均值的方差，其值越小，说明其偏离程度越小，则表明此条路径中链路负载越均衡，所以最终取方差较小的那条路径作为工作路径。这样可以使网络中的资源得到全面合理的利用，提高网络请求的通过率。

2.2 数学定义

在网络算法研究中，实际的 MPLS[6]网络域运用图论抽象为物理拓扑：一个由 V 个节点 E 条边的无向连通加权图 $G(V, E)$ 表示网络（$|V|=n$，$|E|=m$），V 表示网络中 n 个路由器节点的集合，E 表示路由器之间 m 条链路的集合，每条边的链路带宽容量 C 表示能够通过该条边的业务量的最大值。在本算法中，首先给出了如下几个数学定义：

假设网络的链路数为 l，在任意时刻 t，通过≈测量或者其他途径可以得出每条链路的剩余可用带宽 R。

链路带宽利用率 B：对于$\forall(i,j)\in E$，链路（i，j）带宽利用率为：

$$B_i=(1-\frac{R_i}{C_i})\times 100 \quad (1\leqslant i\leqslant l) \tag{1}$$

网络中链路的带宽利用率的均值 E，其数学表达式为：

$$E=\frac{\sum_{i=1}^{l}B_i}{1}\times 100 \quad (1\leqslant i\leqslant l) \tag{2}$$

路径均衡度：就是组成路径的链路带宽利用率相对于网络链路利用率均值的偏差程度，可用数学公式描述(其中 P_l 是所求路径集中的某条路径，N 是 P_l 中链路的条数)：

$$Q_l=\frac{\sum_{i\in P_l}[B_i-E]^2}{N} \tag{3}$$

该算法是以最小化路径的均衡度为目标函数，加之约束条件进行路径的选择。

2.3 构造算法的数学模型

目标：Min（Q(paths(s,d))），paths(s,d)包含于 T(s,d)

约束条件为：

hops(p(s,d))≤ H　　p(s,d) ∈ paths(s,d)

bandwidth(paths(s,d))≥ B　paths(s,d)包含于 T(s,d)

num(paths(s,d))≤ N　　paths(s,d)包含于 T(s,d)

color(p(s,d))包含于 COLOR　　p(s,d) ∈ paths(s,d)

其中，s 表示源节点，d 表示目的节点，p(s,d)表示 s 到 d 的一条路径，T(s,d)表示 s 到 d 的所有路径的集合，paths(s,d)表示 T(s,d)的一个子集。bandwidth(paths(s,d))表示路径 p(s,d)的可预留带宽；num(paths(s,d))表示组成路径集的路径条数。H、B、N 都是常数，分别表示路径的最大长度（即最大跳数）、待转移流量对带宽的要求、所求路径的最多条数；COLOR 表示允许的颜色集合，可以由网络管理员设定。color(p(s,d))表示组成路径 p(s,d)的所有链路的颜色集合。

2.4 算法流程图及复杂度

图 1 为算法流程图。

图 1　算法流程图

算法的关键步骤是入口与出口节点之间的可选路径计算，网络中计算最短路径的复杂度是 $O(n^3)$，计算第二最短路径的复杂度是 $O(n_4)$，计算第 M 最短路径的复杂度是 $O(n^{M+2})$。综合分析算法各步骤，其最终的计算复杂度取决于算法实现中所确定的，需要计算的第 M 条最短路径的复杂度，即 $O(n^{M+2})$。

2.5 LPR 算法

```
createDN（GG，vexnum，arcnum,names,edges）;
                    //图的初始化
PrintGP(names,vexnum);// 界面显示
createUDN(G,vexnum,arcnum,names,edges);
                    //临时中间图初始化用户输入请求；
for( i=0;i<vexnum;i++)
for( j=0;j<vexnum;j++)
{将不满足要求的链路进行裁剪}
u1=sum1/l1;// 计算满足带宽请求的网络拓扑中链路
              的平均利用率，根据平均利用率的取值
              范围对链路进行再次裁剪
```

```
n=KShortestpath(G,city1,city2);// 以裁剪后的网络拓扑
                    图为基础计算出 K 条最短路径
                    // 各路径方差的计算
for(i=0;i<n;i++)
{
t=SizeOf(Paths[i]);
for(j=0;j<t;j++)
{
MemberOf(Paths[i],j,x,y);
sum=sum+(ratio[x][y]-u2)*(ratio[x][y]-u2);
}
Q[i]=sum/t;
}
CopyBNQueue(Paths[k],P);
while(!EmptyQueue(P))// 输出最终路径
{
DeQueue(P,t);
cout<<" "<<" "<<G. vexs[t];
}
cout<<endl<<"\n";     // 修改此路径上各链路的带宽
t=SizeOf(Paths[k]);
for(j=0;j<t;j++)
{
MemberOf(Paths[k],j,x,y);
GG. arcs[x][y]. lwidth=GG. arcs[x][y]. lwidth-B;
GG. arcs[y][x]. lwidth=GG. arcs[x][y]. lwidth;
ratio[x][y]=100-GG. arcs[x][y]. lwidth*100/GG. arcs[x]
[y]. cwidth;
ratio[y][x]=ratio[x][y];
}
```

3 仿真实验

为了证明LPR算法的有效性及其优越性，针对图2所示的网络拓扑进行了两组实验仿真。假设现行网络正在运行，每个节点了解整个网络拓扑和链路状态信息，网络中所有链路均为双向对称链路，链路上的数字表示剩余带宽/最大可预留带宽(×100 kb/s)。

首先，假设连接请求随机产生，其请求范围为随机数50 kb/s~100 kb/s，仿真过程是逐渐增加连接请求数，每次增加10个，仿真内容是实验随着网络中接入连接数的增多，链路的最大带宽利用率的变化情况。本算法的运行结果与XUE算法的比较如图3所示。

由图可见，网络中的连接数少时，两种算法的差别不大，但是随着连接数的增多，最大带宽利用率都在增大，但是在本算法中增加较小。

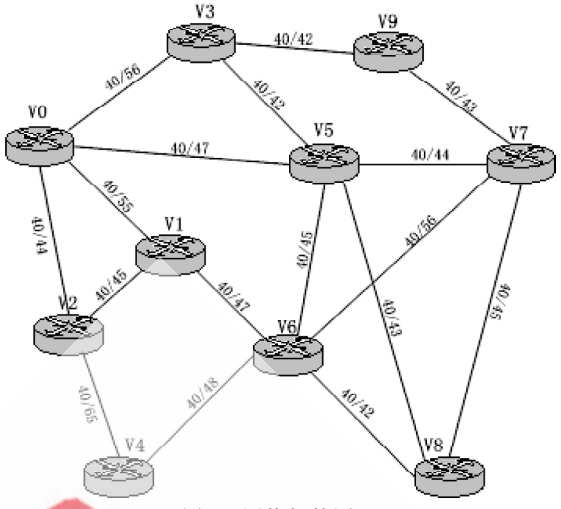

图2 网络拓扑图

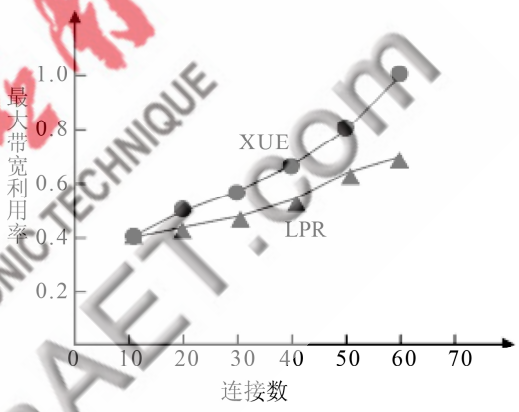

图3 最大带宽利用率随连接数变化情况

另外，在图2所示的网络拓扑图中进行了另一组实验，考察在最短路算法SPF、XUE算法和LPR算法下的路径分布及由此带来的丢包率和带宽利用率等的变化。实验中所使用的数据源示于表1。

表1 仿真中使用的数据源

连接	源	目的	请求带宽
1	1	7	3200 kb/s
2	4	7	2400 kb/s
3	6	9	800 kb/s

在实验中连接请求逐个先后到来，这3个请求在3种算法中选择的路径情况如表2所示。

表2 3种算法的路径选择

算法	连接请求	所选路径
SPF	1	1-6-7
	2	4-6-7
	3	6-7-9
XUE	1	1-6-7
	2	4-6-5-7
	3	6-7-9
LPR	1	1-6-5-7
	2	4-6-7
	3	6-8-7-9

表3是在LPR和XUE两种算法下，链路的利用率状况(为直观只列出所选路径中涉及的链路)。

表3 两种算法下链路利用率

链路	XUE算法下链路利用率/%	LPR算法下链路利用率/%
1-6	83	83
6-7	100	72
4-6	67	67
6-5	65	83
5-7	57	82
7-9	26	26
6-8	5	24
8-7	12	29

图4是在SPF和LPR算法下链路6-7带宽的利用率情况。

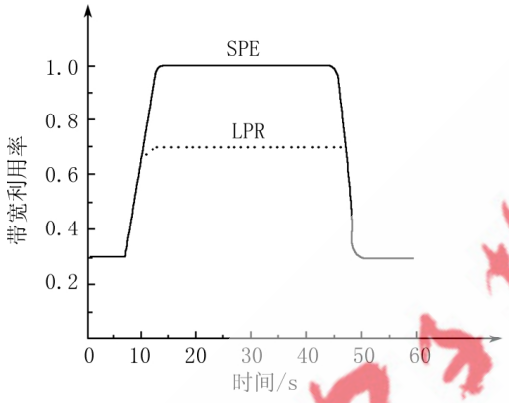

图4 链路6-7在两种算法下的带宽利用率

由仿真实验可知，LPR算法较SPF和XUE算法更加有效：

（1）LPR算法是XUE算法的补充，对于缓解由于流量对资源竞争引起的包丢失具有显著的效果；

（2）LPR算法更加有效地降低了资源利用率，避免瓶颈链路的产生；

（3）LPR算法能更好地调节流量在整个网络上的分布，使网络资源得到均衡分布；

（4）LPR算法大大提高了连接请求的通过率，增加了网络的吞吐量。

参考文献：

[1] 薛希俊，孙雨耕，刘振肖. 基于带宽和跳数的流量工程动态路由选择算法研究[J]. 电子学报，2002，30(2):274-278.

[2] 贾艳萍，孟相如. 基于MPLS流量工程的多路径约束负载均衡方法[J]. 计算机应用，2008，27(3)：522-524.

[3] 朱明英，叶梧. 基于最小干扰机制的MPLS流量工程动态路由算法[J]. 科学技术与工程，2008，8(19)：5394-5398.

[4] HUANG He，LI Wei-Qin. Optimization of MPLS traffic engineering architecture[J]. Journal of Beijing University of Aeronautics and Astronautics，2003，29(3):221-224.

[5] 刘郁恒，张光昭. MPLS流量工程技术的研究[J]. 数据通信，2000(2)：1-4.

[6] WANG Y F，WANG Z. Explicit routing algorithms for internet traffic engineering[A]. IEEE ICCCN'99[C]. Boston，MA. 1999:582-588.

（收稿日期：2008-12-24）

(上接第39页)

程完成如下工作：(1)线程发现发送队列中的有效包句柄则从SRAM的发送队列中将队列头元素取下来；(2)计算每个mpacket在包中的位置，把包从DRAM中以mpacket大小拷贝到TBUF中，其中TBUF是MSF中的发送缓冲区；(3)写入TBUF的单元控制字，表明TBUF包含有效数据；(4)当MSF收到EOP标志的mpacket时，表明该包结束，此后该包将交由外部的MAC设备传输。其过程如图6所示。

一个发送线程一次循环只负责一个mpacket的操作，周而复始。如同接收线程那样，发送线程排好队，如流水线般将发送队列中的元素对应的包，分解为mpacket单元，并逐个按顺序搬运到TBUF缓冲区。

在上述包过滤规则匹配时，微引擎会多次访问DRAM，以及在SRAM中进行搜索。当规则表中有较多规则时，查找规则的算法会变得相当复杂，将严重影响防火墙的处理速度。要使防火墙能快速地完成包过滤功能，可采用2个层次的手段：其一，改进查找算法，比如使用基于状态的动态包过滤算法；其二，充分应用NP内部的并行处理架构，安排好各微引擎工作内容，协调好微引擎内各线程的工作，使NP能高效并行地运行。

参考文献

[1] 宋斌，程勇，刘科全. NP架构千兆线速防火墙的体系结构与关键技术，信息安全与通信保密，2004(8)：22-25.

[2] PANKAJ G，KEOWN M. Algorithms for packet classification. New York:IEEE，March/April 2001：24-32.

[3] DEEPA S，FANG，Wu Chang. Performance analysis of multi-dimensional packet classification on programmable network processors. New York： IEEE，Proceeding of the 29th Annual IEEE International Coference on Local Computer Networks (LCN'04).

[4] ERIK J J，AARON R K. IXP2400/2800 Programming. INTEL PRESS.

(收稿日期：2008-12-24)