软件学报 ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*,2018,29(12):3904−3920 [doi: 10.13328/j.cnki.jos.005426]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

# LFA 算法的一种高效实现方法*

耿海军[1,2], 施新刚[3], 王之梁[3], 尹 霞[4], 尹少平[1]

[1](山西大学 软件学院,山西 太原 030006)

[2](网络与交换技术国家重点实验室(北京邮电大学),北京 100876)

[3](清华大学 网络科学与网络空间研究院,北京 100084)

[4](清华大学 计算机科学与技术系,北京 100084)

通讯作者: 耿海军, E-mail: ghj123025449@163.com

**摘 要**: 研究表明,网络中的故障不可避免而且频繁出现.当故障发生时,目前互联网部署的域内路由协议需要经历收敛过程.在此过程中,路由信息可能不一致,从而导致报文丢失,降低了路由可用性.因此,业界提出了利用LFA(loop free alternates)应对网络中发生的单故障情形,从而提高路由可用性.然而,已有的 LFA 实现方式算法时间复杂度大,需要消耗大量的路由器 CPU 资源.针对该问题严格证明了当网络中出现单故障时,只需要为特定的节点计算备份下一跳,其余受该故障影响节点的备份下一跳和该特定节点的备份下一跳是相同的.基于上述性质,分别讨论了对称链路权值和非对称链路权值中对应的路由保护算法.实验结果表明:与 LFA 相比较,该算法的执行时间降低了 90%以上,路径拉伸度降低了 15%以上,并且与 LFA 具有同样的故障保护率.

**关键词**: 网路故障;IP 快速重路由;路由保护;路径拉伸度;故障保护率

**中图法分类号**: TP393

中文引用格式: 耿海军,施新刚,王之梁,尹霞,尹少平.LFA 算法的一种高效实现方法.软件学报,2018,29(12):3904−3920. http://www. jos.org.cn/1000-9825/5426.htm

英文引用格式: Geng HJ, Shi XG, Wang ZL, Yin X, Yin SP. Efficient implementation method for LFA. Ruan Jian Xue Bao/ Journal of Software, 2018,29(12):3904−3920 (in Chinese). http://www.jos.org.cn/1000-9825/5426.htm

## Efficient Implementation Method for LFA

GENG Hai-Jun[1,2], SHI Xin-Gang[3], WANG Zhi-Liang[3], YIN Xia[4], YIN Shao-Ping[1]

[1](School of Software Engineering, Shanxi University, Taiyuan 030006, China)

[2](State Key Laboratory of Networking and Switching Technology (Beijing University of Posts and Telecommunications), Beijing 100876, China)

[3](Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing 100084, China)

[4](Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

**Abstract**: Lots of related researches have shown that network failures occur inevitably and frequently on the Internet. When network failures occur, the currently deployed intra-domain routing protocol need to re-convergent. During the process of re-convergence, the packets may be lost due to inconsistent routing information, thus greatly reducing the Internet routing availability. LFA was proposed to

* 基金项目: 国家自然科学基金(61702315, 61402253, 61872226); 网络与交换技术国家重点实验室(北京邮电大学)开放课题(SKLNST-2018-1-19); 国家高技术研究发展计划(863)(2015AA015603, 2015AA016105);

Foundation item:; National Natural Science Foundation of China (61702315, 61402253, 61872226); Open Foundation of State Key Laboratory of Networking and Switching Technology(Beijing University of Posts and Telecommunications) (SKLNST-2018-1-19); National High Technology Research and Development Program of China (863) (2015AA015603, 2015AA016105)

收稿时间: 2016-11-01; 修改时间: 2017-01-03; 采用时间: 2017-02-15; jos 在线出版时间: 2018-03-13

CNKI 网络优先出版: 2018-03-13 17:17:59, http://kns.cnki.net/kcms/detail/11.2560.TP.20180313.1717.003.html

cope with all the single failure scenarios. However, the existing LFA implementation algorithms are time-consuming and require a large amount of router CPU resources. This paper proves that when a single fault occurs in a network, it only needs to calculate the backup next hop for a specific node. The backup next hop of all the other affected nodes by the fault is the same as that of the specific node. Based on the above property, the paper proposes two routing protection algorithms to provide protection for networks with symmetric and asymmetric. The results show that these approaches reduce more than 90% computation overhead compared to LFA, and achieve less than 15% path stretch. Moreover, they can provide comparable protection ratio with LFA.

**Key words**: network failure; IP fast re-route; routing protection; path stretch; protection ratio

互联网最初的原型是美国国防部高级研究计划局设计的阿帕网(advanced research project agency network,简称 APRANET),仅仅由 4 台主机组成,发展到今天,仅仅用了不到 50 年的时间,其发展速度远远超出了人们的想象[1,2].随着互联网的快速发展和规模的逐渐扩大,其应用已经延伸到人们的工作、生活、学习和娱乐等各个领域,并且成为全球最大的通信系统.正如微软首席官 Ozzie 所述,我们生活在以网络为中心的时代[3].

随着互联网的发展,其应用范围呈现出了显著的变化.最初,互联网主要支持一些非实时应用,如电子邮件、传输文件等.然而,如今大量的实时业务[4−6],如 IP 语音(voice over Internet protocol,简称 VoIP)、股票在线交易、远程手术、视频流媒体和即时通信等,在互联网上大量传播,这些新型应用对路由可用性[7,8]提出了更高的要求.因此,路由可用性将直接影响用户的财产安全甚至生命安全.路由可用性是指用户能够得到请求服务的概率,可以表示为 MTTF/(MTTF+MTTR),其中,MTTF(mean time to failure)表示平均故障间隔时间,MTTR(mean time to repair)表示故障平均修复时间.从上述定义可以知道,可以通过增加 MTTF 的值和减少 MTTR 的值来提升路由可用性.然而,只能通过提高硬件设备的质量来增加 MTTF 的值,这将会给 ISP 带来额外的负担.因此在实际中,一般通过减少 MTTR 的值来提高路由可用性.

相关研究表明:网络中的故障频繁发生[9,10],并且不可避免.在故障修复期间,路由协议需要经历一段时间的收敛过程,在此过程中,将会出现路由黑洞、路由环路[11,12]等现象,从而导致网络中断,报文丢失,降低了路由可用性.当一条 OC-48 的链路断开 10s,将导致 300 万个大小为 1KB 的报文丢失[13].当故障发生时,实时应用通常要求毫秒级的故障修复时间,然而目前互联网部署的域内路由协议,如 OSPF 和 IS-IS,需要几秒甚至几十秒的时间来完成收敛,目前互联网部署的路由协议无法满足实时应用对路由可用性的要求[14,15].因此,如何提高域内路由可用性[16,17],成为互联网亟待需要解决的一个问题.

针对互联网路由的可用性问题,越来越多的科研工作者开始投身到研究如何提升网络快速应对故障的能力[18−21].提高路由可用性的方案[22]可以分为两类:被动恢复方案和路由保护方案.其中,被动恢复方案主要研究当网络出现故障时,如何加快路由收敛速度,尽量减少网络中断时间.然而当网络中的链路频繁断开时,该方案可能导致路由不稳定.因此,该方案必须权衡路由收敛速度和路由稳定性.与被动恢复方案相比较,路由保护方案更受学术界青睐.路由保护方案的基本思想是:根据相关规则事先计算备份路由,当网络中出现故障时,利用事先计算的备份路由转发数据包,从而最大化减少报文丢失,缩短网络服务中断时间[23].

根据报文转发方式,可以将路由保护方案分为两种方式:逐跳转发和非逐跳转发.基于逐跳转发的路由保护方案和目前互联网部署的域内路由协议的转发方式是一致的,容易部署,因此受到工业界和学术界的关注;而非逐跳转发方式需要借助辅助机制,如 Not-Via[24]、隧道技术(tunnel)[25]、多协议标签交换(multi-protocol label switching,简称 MPLS)[26]来转发报文,该机制部署复杂,给路由器带来了一定的负担.并且许多 ISP 希望在纯 IP 网络中实现路由保护方案,以减少网络配置,降低运营成本,因此,本文研究基于逐跳转发方式的路由保护方案.LFA(loop-free alternate)[27]是一种典型的路由保护方案,该方案采用逐跳转发方式,实现简单,容易部署,是现在互联网唯一部署的域内路由保护方案.然而,LFA 的实现方式存在以下两个方面的问题.

(1) 为了实现 LFA,计算节点需要构造多棵最短路径树.相关研究表明,构造最短路径树需要消耗大量的计算资源[28−30].因此,LFA 算法复杂度过高,并且算法复杂度随着网络节点度数的增加而增加[31],降低了路由器的性能;

(2) 没有明确规定如何从所有可选下一跳中选择哪个作为最终的备份下一跳,如果采用随机选择算法,则

可能导致重路由路径的拉伸度过大,从而浪费大量的网络带宽资源.

因此,如何降低 LFA 方案的算法时间复杂度[31]和重路由路径的拉伸度,是一个重要的研究问题.在文献[31]中,相关作者提出利用 TBFH 算法来降低 LFA 中链路保护方案实现方式的复杂度.研究表明,该方案降低了 LFA 方案中链路保护方案实现方式的复杂度.然而,该方案具有下面几个缺点:(1) 没有考虑 LFA 中节点保护条件;(2) 故障保护率较低;(3) 没有考虑重路由路径的拉伸度;(4) 没有考虑非对称权值网络中 TBFH 的实现方式.针对已有研究存在的问题,本文提出了一种轻量级的基于逐跳转发方式的路由保护方案,该算法不仅具有较小的时间复杂度,并且重路由路径具有较小的路径拉伸度,同时可以提供和 LFA 同样的故障保护率.本文通过严格推理得出如下结论:当网络中出现单故障时,只需要为特点节点计算备份下一跳,而其余受该故障影响的节点的备份下一跳和该特定节点的备份下一跳是相同的.基于该性质,分别讨论了对称链路权值网络和非对称链路权值网络对应的算法.当网络中的链路权值对称时,提出了一种线性时间复杂度的路由保护方案,该算法的时间复杂度为 $O(2\cdot E+V)$,超越了已有的所有算法的计算性能.相反,当网络中的链路权值不对称时,本文提出的算法的计算时间远远小于构造一棵最短路径树的时间.

本文第 1 节介绍相关工作.第 2 节是问题描述.第 3 节和第 4 节分别介绍对称权值网络和非对称权值网络对应的路由保护算法.第 5 节对算法进行讨论.第 6 节通过实验模拟并且评价算法.第 7 节是结论和下一步研究方向.

## 1 相关工作

互联网在设计之初就非常重视路由可用性,设计了动态路由协议来应对网络故障.对于目前互联网部署的域内路由协议,单链路故障或者单节点故障就可导致全网的路由重计算和路由收敛.在路由收敛过程中,各个路由器都要重新计算路由,因此可能导致报文丢失.互联网部署的域内路由协议收敛时间过长,无法满足实时应用对网络时延的要求.因此,学术界和工业界提出利用路由保护方案来提高路由可用性.根据工作方式的不同,可以将路由保护方案分为被动恢复方案和路由保护方案,其中:被动恢复方案主要关注当网络出现故障的时候,如何加快路由收敛速度;主动保护方案通过预先计算备用路径,当网络出现故障的时候,检测到故障的结点立即启用备用路径,从而可以有效减少网络中断的时间.下面将分别介绍这两种方案.

### 1.1 被动恢复方案

(1) 调整域内路由协议参数

文献[28–30]通过调整域内路由协议收敛过程中各个阶段的参数来加快路由收敛过程.

- 故障检测阶段,加快 Hello 包的发送频率,从而加快故障检测的速度;
- 故障通知阶段,动态调整链路状态广播(link-state advertisement,简称 LSA)发送时间,加快故障传播速度;
- 路由计算阶段,采用增量最短路径优先算法(incremental shortest path first,简称 i-SPF)[35]加快最短路径树的更新时间.

该方案通过调整域内路由协议的默认参数来加快路由收敛速度,实现简单,容易部署.然而该方案容易引起路由震荡,从而影响路由稳定性,并且给路由器带来了额外的负担,增加了网络开销.

(2) 修改链路权值

链路权值在路由的计算中扮演着重要的角色,网络管理员可以通过修改链路权值来解决链路拥塞和路由环路等问题.当网络管理员需要关闭某条链路时,如果直接将其代价设置为无穷大,则可能引起路由环路,从而造成报文丢失.因此,作者在文献[36]中提出无环路收敛条件(loop free convergence,简称 LFC)来处理该问题.作者证明:如果按照一定的权值序列修改某条链路的代价,则路由收敛过程将不会出现路由环路.因此,最终可以将该链路关闭.然而,该方案只适合预知的链路故障,无法适应突发故障.

### 1.2 路由保护方案

(1) ECMP

等价多路径路由(equal cost multiple paths,简称 ECMP)[37]既是最简单也是最早使用的路由保护方案.如果源节点到目的节点有多条路径具有相同的最小代价,则可以将其作为备份路径.该方案实现简单,容易部署,然而要求备份路径必须具有相同的最小代价,因此,ECMP 方案对路由可用性的贡献并不是很大.相关研究通过调整链路权值来配置相同代价的最短路径,然而该问题被证实是 NP-Hard 问题[38].

(2) IPFRR

IETF 提出利用 IP 快速重路由框架(IP fast re-route,简称 IPFRR)[27]来缓解因网络故障造成的报文丢失率,尽量缩短网络中断时间.LFA 是基于 IPFRR 框架提出的一种解决方案,该方案实现简单,因此得到了路由器厂商的支持,是现在唯一部署域内路由保护方案.在 RFC5286 中,IETF 发布了 IPFRR 的基本框架,提出了 LFA 的实现形式,其中包括单链路保护条件(loop free condition,简称 LFC)、单节点保护条件(node protection condition,简称 NPC)和并发故障保护条件(downstream condition,简称 DC).然而,已有的针对 LFA 的实现方式存在两个方面的问题:① 算法复杂度高,每个路由器需要为其邻居节点构造最短路径树,算法复杂度随着网络中节点度数的增加而增加,因此实现代价较大;② 没有规定如何从可选下一跳中选择最终备份下一跳,因此造成重路由路径拉伸度较大.相关研究[32]表明,LFA 的单故障保护率在 50%左右.文献[33]通过调整链路权值进一步提高 LFA 的故障保护率,实验结果表明:在大多数网络图中,该方案都可以得到较好的结果;然而对于某些稀疏图,该方案未能达到理想的效果.针对该问题,文献[34]研究了网络拓扑和 LFA 故障保护率之间的关系,通过修改网络拓扑结构来增加 LFA 的故障保护率.实验结果表明:该方案只需要在网络中增加极少数的边,就可以大幅度提高 LFA 的故障保护率.本文提出的方案可以在文献[33,34]的基础上执行,因此利用该技术可以大大提高本文方案的故障保护率.

(3) 路由偏转

基于 IPFRR 的基本思想,作者提出利用路由偏转[39]方案来提高路由可用性.路由偏转的核心思想是:首先,利用无环路规则计算源节点到目的节点的所有可选下一跳;其次,利用标签技术实现报文的灵活转发.虽然该方案可以提高路由可用性,但是该方案实现复杂,开销较大,难以实际部署.

(4) MRC

多配置路由(multiple routing configurations,简称 MRC)[40]的基本思想是:每个路由器保存多个配置图,每个配置包括所有的节点和链路,通过调整链路权值,从而使得部分节点和链路在该配置中得到保护,最终构造出针对所有可能出现的单故障的配置图.当报文在转发过程中遇到故障时,可以利用事先配置好的路由转发该报文.然而在现实网络中,该方案需要较多的配置,需要消耗大量的计算资源和存储开销.

(5) FCP

FCP(failure carrying packet)[41]的基本思想是:将报文在转发过程中遇到的故障信息存储在该报文的头部,当报文到达某个节点时,该节点首先检测该报文头部的故障信息字段,根据该字段构造新的拓扑结构;然后,利用新的拓扑结构重新计算最短路径,从而实现报文的无环路转发.该方案最大的好处是消除了路由收敛过程,然而计算复杂度高,对路由协议的改动比较大,不容易实际部署.

(6) 基于 Not-Via 地址的快速重路由

文献[42]提出了基于 Not-Via 地址的快速重路由方案.该机制使用特殊地址 Not-Via 显示表明网络中的故障,在转发报文的过程中有效避开该故障.当报文在转发过程中遇到故障时,该报文将会被封装成特殊形式的报文,然后转发到合适的中转节点,最后,中转节点对该报文解封装,并且按照最短路径转发该报文.然而,该方案需要辅助地址的协助,计算开销和存储开销较大,不支持增量部署,因此很难得到 ISP 的支持.

(7) MPLS

利用 MPLS 可以快速、灵活转发数据包,它为网络管理员提供了便捷的配置方式.然而 MPLS 的控制和管理开销较大,尤其当标签交换路径(LSP)的数量较多的时候.相关研究表明[42]:目前,互联网中已经部署 MPLS 的 AS 数量仅仅在 7%左右.

## 2 网络模型和问题描述

### 2.1 网络模型

本节将描述网络模型,并且在表 1 中列出了本文用到的所有符号.网络可以表示为带权有向图 $G=(V,E)$,其中,$V$ 表示网络中路由器(节点)的集合,$E$ 表示网络中链路的集合.对于网络中任意一条链路$(i,j)\in E$,$w=(i,j)$表示该链路的代价,对于网络中任意节点$\forall x\in V$,$neighbor(x)$表示节点 $x$ 的邻居集合.对于网络中任意两个节点$\forall x,y\in V$,$cost(x,y)$表示节点 $x$ 到节点 $y$ 的最短路径的代价.对于任意节点$\forall v\in V$,$spt(v)$表示以节点 $v$ 为根的最短路径树,包含了节点 $v$ 到达其余节点的最短路径.在最短路径树 $spt(v)$中,对于该树中的任意节点$\forall u\in V$,$subtree(v,u)$表示在 $spt(v)$中以节点 $u$ 为根的子树中节点的集合,$child(v,u)$表示在 $spt(v)$中,节点 $u$ 的孩子节点的集合.假设源地址为 $s$,目的地址为 $d$,$bestn(s,d)$表示节点 $s$ 到节点 $d$ 的最优下一跳,$backn(s,d)$表示节点 $s$ 到节点 $d$ 的备份下一跳,$sp(s,d)$表示节点 $s$ 到节点 $d$ 的最短路径.对于任意节点$\forall v\in V$,$rspt(v)$表示以节点 $v$ 为根的反向最短路径树,即:以节点 $v$ 为根的汇聚树,包含了所有节点到达节点 $v$ 的最短路径.对于该树中的任意节点$\forall u\in V$,$rsubtree(v,u)$表示在 $rspt(v)$中以节点 $u$ 为根的子树中节点的集合.

**Table 1** Notations

**表 1** 本文用到的符号

| 符号 | 含义 |
|---|---|
| $G=(V,E)$ | 网络拓扑 |
| $w=(i,j)$ | 链路$(i,j)\in E$ 的代价 |
| $neighbor(x)$ | 节点 $x$ 的邻居节点 |
| $cost(x,y)$ | 节点 $x$ 到节点 $y$ 的最短路径的代价 |
| $spt(v)$ | 以节点 $v$ 为根的最短路径树 |
| $bestn(s,d)$ | 节点 $s$ 到节点 $d$ 的最优下一跳 |
| $backn(s,d)$ | 节点 $s$ 到节点 $d$ 的备份下一跳 |
| $sp(s,d)$ | 节点 $s$ 到节点 $d$ 的最短路径 |
| $subtree(v,u)$ | 在 $spt(v)$中以节点 $u$ 为根的子树中节点 |
| $child(v,u)$ | 在 $spt(v)$中节点 $u$ 的孩子节点 |
| $rspt(u)$ | 以 $u$ 为根的反向最短路径树 |
| $rsubtree(v,u)$ | 在 $rspt(v)$中以节点 $u$ 为根的子树中节点 |

### 2.2 问题描述

目前,互联网部署的域内路由协议如 OSPF,每个节点根据链路状态数据库中的拓扑信息构造以自身为根的最短路径树,根据该树计算出到所有目的的最优下一跳.当节点到目的的最优下一跳出现故障时,发往该目的地址的报文将会被丢弃.为了灵活应对网络中的突发故障,IETF 在 RFC5286 中发布了 LFA 标准,其中包括无环路条件(loop free condition,简称 LFC)、节点保护条件(node protection condition,简称 NPC).

- LFC:假设目的地址为 $d$,如果$(s,u)\in sp(s,d)$,当链路$(s,u)$出现故障时,节点 $s$ 可以将报文转发给其邻居 $m\in neighbor(s)$,当且仅当满足 $cost(m,d)<cost(m,s)+cost(s,d)$;
- NPC:假设目的地址为 $d$,如果$(s,u)\in sp(s,d)$,当节点 $u$ 出现故障时,$s$ 可以将报文转发给其邻居 $m\in neighbor(s)$,当且仅当满足 $cost(m,d)<cost(m,u)+cost(u,d)$.

为了实现 LFC,节点 $s$ 需要获得 $cost(m,d)$,$cost(m,s)$和 $cost(s,d)$的具体数值,其中,$cost(s,d)$可以从 $spt(s)$中得出,而 $cost(m,d)$需要从 $spt(m)$中得出.因此,为了获得 $cost(m,s)$和 $cost(m,d)$的数值,节点 $s$ 需要构造一棵以 $m$ 为根的最短路径树.当节点 $s$ 有 $k$ 个邻居节点时,为了获得其所有邻居到目的的最小代价,节点 $s$ 需要构造 $k$ 棵最短路径树.从上面的分析可知,实现 LFC 的复杂度与运行算法的节点的度数密切相关.因此,该实现方式的扩展性较差.因为 NPC 的实现方式和 LFC 的实现方式是相同的,所以不再详细阐述.

下面举例说明已有的 LFC 实现方式.图 1 表示以节点 $s$ 为根的最短路径树,其中,实线表示该树上的链路,虚线则表示不在该树上的链路.为了实现 LFC,节点 $s$ 需要构造以其邻居节点($u$ 和 $h$)为根的最短路径树;为了实现 NPC,节点 $s$ 需要构造以其邻居节点的孩子节点($b$,$c$,$i$ 和 $j$)为根的最短路径树.因此,该实现方式的复杂度随着

网络节点度数的增加而增加,不利于 LFA 方案的实际部署.

因此,为了降低 LFA 算法的复杂度,本文首先提出下面定理.定理 1 给出了如何为子树 *subtree*(*s*,*u*)中的所有节点计算备份下一跳,根据该定理可知,该子树中所有节点的备份下一跳和节点 *u* 的备份下一跳是相同的.根据定理 1 可知:在 *spt*(*s*)中,为了保护链路(*s*,*u*)和(*s*,*h*),只需要为节点 *u* 和 *h* 计算备份下一跳.同样,为了保护节点 *u* 和 *h*,只需要为节点 *b*,*c*,*i* 和 *j* 计算备份下一跳.因此,为了保护网络中的链路和节点,算法只需要为特定的节点计算备份下一跳即可,完全没有必要为网络中所有节点计算备份下一跳,这将降低算法的时间复杂度,减轻路由器的负担.下面将在两种类型的网络中分别讨论如何为上述这些特定节点计算备份下一跳.

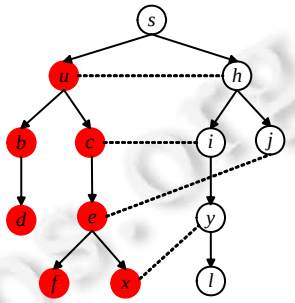


Fig.1 Shortest path tree rooted at *s*

图 1 以节点 *s* 为根的最短路径树

**定理 1**. 如果 *backn*(*s*,*u*)=*n*,对于任意节点如果 *m*∈*subtree*(*s*,*u*),则 *n*∈*backn*(*s*,*m*).

证明:因为 *backn*(*s*,*u*)=*n*,所以,

$$cost(n,s)+cost(s,u)>cost(n,u) \tag{1}$$

由公式(1)可得:

$$cost(n,s)+cost(s,u)+cost(u,m)>cost(n,u)+cost(u,m) \tag{2}$$

由于 *m*∈*subtree*(*s*,*u*),可知:

$$cost(s,m)=cost(s,u)+cost(u,m) \tag{3}$$

根据公式(2)和公式(3)可得:

$$cost(n,s)+cost(s,m)>cost(n,u)+cost(u,m) \tag{4}$$

因为,

$$cost(n,u)+cost(u,m)\geqslant cost(n,m) \tag{5}$$

所以由公式(4)和公式(5)可得 *cost*(*n*,*s*)+*cost*(*s*,*m*)>*cost*(*n*,*m*),即 *n*∈*backn*(*s*,*m*).定理得证. □

## 3 对称链路权值下的路由保护方案

当网络中链路权值对称时,即:对于任意链路(*i*,*j*)∈*E*,都有 *w*=(*i*,*j*)=*w*=(*j*,*i*)成立;同时,对于网络中任意两个节点∀*x*,*y*∈*V*,都有 *cost*(*x*,*y*)=*cost*(*y*,*x*).利用上述两个性质,可以优化 LFC 和 NPC 的实现方式,从而设计出高效的算法.下面将分别介绍链路保护算法和节点保护算法.

### 3.1 链路保护条件

根据定理 1 可知:在 *spt*(*s*)中,当链路(*s*,*u*)∈*spt*(*s*)时,为了保护链路(*s*,*u*),节点 *s* 需要为节点 *u* 计算备份下一跳.下面的定理给出了计算备份下一跳的一个重要性质,利用该性质可以降低算法复杂度.

**定理 2**. 当网络中链路的权值对称时,如果(*s*,*u*)∈*sp*(*s*,*u*)并且 *backn*(*s*,*u*)≠∅,则必定存在一条链路(*x*,*y*),从而使得 *x*∈*subtree*(*s*,*u*),*y*∉*subtree*(*s*,*u*),*bestn*(*s*,*y*)∈*backn*(*s*,*u*)同时成立.

定理 2 提供了如何为节点 *u* 计算备份下一跳的方法:遍历 *subtree*(*s*,*u*)中的节点并且检测其邻居节点,这些邻居节点必须满足不在 *subtree*(*s*,*u*)中并且满足链路保护的条件.该条件将在定理 3 中给出.在图 1 中,如果

$backn(s,u)\neq\varnothing$,为了计算节点 $u$ 的备份下一跳,遍历 $subtree(s,u)$中的所有节点,必定能找到一条链路$(x,y)$,使得 $x\in subtree(s,u)$,$y\notin subtree(s,u)$和 $h\in backn(s,u)$同时成立,即,节点 $u$ 的备份下一跳为 $h$.当链路$(s,u)$正常工作时,节点 $s$ 到 $subtree(s,u)$中所有节点的最优下一跳为 $u$;当链路$(s,u)$出现故障时,节点 $s$ 到 $subtree(s,u)$中所有节点的备份下一跳为 $h$.为了证明定理 2 的正确性,首先证明几个引理.

**引理 1**. 当网络中链路的权值对称时,如果$(s,u)\in sp(s,u)$,则 $subtree(s,u)\cap subtree(u,s)=\varnothing$.

证明:下面将使用反证法来证明该定理.假设 $subtree(s,u)\cap subtree(u,s)=v$.因此,一方面在 $spt(u)$中,$s\in sp(v,u)$;另一方面在 $spt(s)$中,因为 $u$ 是 $s$ 的孩子节点,则 $s\notin sp(v,u)$,因此得出矛盾.即假设不成立.定理得证. □

**引理 2**. 当网络中链路的权值对称时,$V–subtree(s,u)=subtree(u,s)+V'$,其中,$V'$为网络中剩余节点.

证明:根据引理 1 可知 $subtree(s,u)\cap subtree(u,s)=\varnothing$,网络中所有节点可以分为 3 种类型,一部分节点包含在 $subtree(s,u)$,另外一部分包含在 $subtree(u,s)$中,其余为网络中剩余节点,因此,网络中节点可以表示为

$$V–subtree(s,u)=subtree(u,s)+V'.$$ □

**引理 3**. 当网络中链路的权值对称时,对于节点 $p\in V$,如果 $s\in sp(p,u)$,则 $s\in sp(bestn(s,p),u)$.

证明:令 $bestn(s,p)=m$.根据 $s\in sp(p,u)$,可以得到:

$$cost(p,u)=cost(p,s)+cost(s,u) \tag{6}$$

由于 $m$ 是 $s$ 到 $p$ 的最优下一跳,因此:

$$cost(p,s)=cost(p,m)+cost(m,s) \tag{7}$$

根据公式(6)和公式(7)可知,$cost(p,u)=cost(p,m)+cost(m,s)+cost(s,u)$.

因此,$cost(m,u)=cost(m,s)+cost(s,u)$.即,$s\in sp(bestn(s,p),u)$. □

下面给出定理 2 的详细证明过程.

证明定理 2:

首先证明节点 $y\in V'$.用反证法来证明.假设 $y\notin V'$,根据引理 2 可知,$y\in subtree(s,u)$,因此 $s\in sp(y,u)$,根据引理 3,$s\in sp(bestn(s,y),u)$,与 $bestn(s,y)\in backn(s,u)$矛盾.因此,$y\in V'$.

下面使用反证法来证明该定理.假设不存在任何链路$(x,y)$使 $x\in subtree(s,u)$和 $y\notin subtree(s,u)$且 $bestn(s,y)\in backn(s,u)$同时成立.即:对于任意节点$y$,与该节点相连的另外一端只能在集合 $V–subtree(s,u)$中,根据引理 2 可知,$V–subtree(s,u)=subtree(u,s)+V'$并且 $y\in V'$,因此 $sp(y,u)$必经过 $subtree(u,s)$中的节点,根据链路对称可知 $s\in sp(y,u)$,根据引理 3 可知 $s\in sp(bestn(s,y),u)$,与 $bestn(s,y)\in backn(s,u)$矛盾.因此,定理得证. □

**定理 3**. 当网络中链路的权值对称时,节点 $bestn(s,y)\in backn(s,u)$成立的充分条件是以下 3 个条件同时成立:

(1) $(s,u)\in sp(s,u)$;

(2) 存在一条链路$(x,y)$,从而使得 $x\in subtree(s,u)$且 $y\notin subtree(s,u)$且 $y\neq s$;

(3) $2\cdot cost(s,u)>cost(s,x)+cost(x,y)+cost(s,y)–2\cdot cost(s,bestn(s,y))$

证明:该定理中的前两个条件已经在定理 2 中给出了证明.下面分析条件(3).

令 $bestn(s,y)=m$,因此得到

$$cost(s,y)=cost(s,m)+cost(m,y) \tag{8}$$

由于 $x\in subtree(s,u)$,得出:

$$cost(s,x)=cost(s,u)+cost(u,x) \tag{9}$$

将公式(8)和公式(9)带入公式条件(3)中,可以得到:

$$cost(m,s)+cost(s,u)>cost(m,y)+cost(y,x)+cost(x,u) \tag{10}$$

由于 $cost(m,y)+cost(y,x)+cost(x,u)\geqslant cost(m,u)$,所以 $cost(m,s)+cost(s,u)>cost(m,u)$.

因此 $s\notin sp(m,u)$,即 $m\in backup(s,u)$.因此,该定理成立. □

定理 3 给出了为节点 $u$ 计算备份下一跳的充分条件,前两个条件已经在定理 2 中给出了证明.第 3 个条件是节点间必须满足的定量关系,该条件中的所有变量值都可以从链路状态数据库和 $spt(s)$中得到.因此,根据该公式很容易判断节点 $bestn(s,y)$是否满足备份下一跳.

### 3.2 节点保护条件

在 $spt(s)$中,如果链路$(s,u)\in spt(s)$,当节点 $u$ 出现故障时,节点 $s$ 需要为 $child(s,u)$中的节点计算备份下一跳.由此可知:为了保护某条链路,节点 $s$ 只需要为一个节点计算备份下一跳.然而为了保护某个节点,节点 $s$ 需要为该节点的所有孩子节点计算备份下一跳.

定理 4 提供了如何为节点 $u$ 的孩子节点 $v$ 计算备份下一跳的方法.定理 5 给出了节点保护条件.这两个定理和链路保护条件中的定理相似,因此不再具体说明.

**定理 4**. 当网络中链路的权值对称时,如果$(s,u)\in sp(s,v)$,$(u,v)\in sp(s,v)$,$backn(s,v)\neq\varnothing$,则必定存在一条链路$(x,y)$,使得 $x\in subtree(s,v)$,$y\notin subtree(s,u)$和 $bestn(s,y)\in backn(s,v)$同时成立.

**定理 5**. 网络中链路的权值对称时,节点 $bestn(s,y)\in backup(s,v)$成立的充分条件是以下 3 个条件同时成立:

(1) $(s,u)\in sp(s,v)$并且$(u,v)\in sp(s,v)$;

(2) 存在一条链路$(x,y)$,从而使得 $x\in subtree(s,v)$且 $y\notin subtree(s,u)$且 $y\neq s$;

(3) $2\cdot cost(s,v)>cost(s,x)+cost(x,y)+cost(s,y)-2\cdot cost(s,bestn(s,y))$.

### 3.3 链路保护算法

算法 1 描述了如何为节点 $u$ 计算备份下一跳:首先,将节点 $s$ 和 $subtree(s,u)$中的所有节点标记为红色(算法 1 第 1 行、第 2 行);遍历子树 $subtree(s,u)$中的所有节点(算法 1 第 4 行);对于该子树中的任意节点 $x$,访问其每一个邻居节点 $y$,判断其是否满足定理 3 中的条件(算法 1 第 6 行~第 9 行);在找到的所有满足条件的节点 $y$ 中,选择保护路径最短的一个作为最终节点(算法 1 第 10 行);最后,算法返回节点 $u$ 的备份下一跳.

**算法 1**. SynLinkProtection.

Input: $SPT(s)$, $u$, $G=(V,E)$;

Output: $backn(s,u)$.

```
1:  将 subtree(s,u)中所有节点标记为红色;
2:  将节点 s 标记为红色
3:  min←∞
4:  For x∈subtree(s,u) do
5:    For y∈neighbor(x) do
6:      If 节点 y 是红色的 or 定理 3(3)不成立 or cost(s,y)+w(x,y)−cost(s,u)+cost(s,x)≥min
7:        continue
8:      EndIf
9:      backn(s,u)←backn(s,y)
10: min←cost(s,y)+w(x,y)+cost(s,x)−cost(s,u)
11:   EndFor
12: EndFor
13: return backn(s,u);
```

算法 1 描述了节点 $s$ 如何为与其直连的节点 $u$ 计算备份下一跳的过程.然而,为了保护所有与节点 $s$ 直接相连的链路,节点 $s$ 需要为其直连的所有节点计算备份下一跳,这将需要运行多次算法 1 来实现.下面来分析算法的复杂度.

**定理 6**. 当网络中链路的权值对称时,节点 $s$ 为所有与其直连的节点计算备份下一跳的时间复杂度为 $O(2\cdot E+V)$.

证明:假设节点 $s$ 有 $m$ 个邻居节点,则为了保护与其直接相连的所有链路,需要运行 $m$ 次算法 1.在运行 $m$ 次算法 1 的过程中,除去节点 $s$ 外,网络中每个节点最多被访问一次,而每条链路则最多被访问两次,因此算法的时间复杂度为 $O(2\cdot E+V)$. □

### 3.4 节点保护算法

根据定理 4 可知:为了实现节点保护算法,只需要将链路保护算法中所有的变量 $u$ 改为 $v$ 即可.节点保护算法用 SynNodeProtection 表示.因此,该变化将不会影响算法的时间复杂度.因此,节点保护算法的复杂度仍然是 $O(2\cdot E+V)$.

## 4 非对称链路权值下的路由保护方案

以上讨论了对称链路权值网路中的链路保护算法和节点保护算法.当网络中链路的权值不对称时,将不能采用上述算法来解决该问题.这是因为上述定理 2~定理 5 都是在对称链路权值的条件下才成立的,在非对称链路权值条件下,上述定理不再成立,因此当网络中链路的权值不对称时,需要设计一种新的方法来解决该问题.

### 4.1 链路保护条件

根据定理 1 可知:在 $spt(s)$中,当链路$(s,u)\in spt(s)$时,为了保护链路$(s,u)$,节点 $s$ 需要为节点 $u$ 计算备份下一跳.定理 7 给出了计算链路保护的方法,根据定理 7 可知:如果某个节点是 $s$ 的邻居节点,并且该节点到节点 $u$ 的最短路径不经过 $s$,则该节点可以作为节点 $u$ 的备份下一跳.

**定理 7**. 如果$(s,u)\in sp(s,u)$,$backn(s,u)\neq\varnothing$成立,则集合 $neighbor(s)\cap(V-rsubtree(s))$中的所有节点都可以作为节点 $u$ 的备份下一跳,即 $neighbor(s)\cap(V-rsubtree(s))\subset backn(s,u)$.

证明:利用反证法来证明该定理.假设 $m\in neighbor(s)\cap(V-rsubtree(s))$,但是该节点 $m\notin backn(s,u)$,则节点 $m$ 到 $u$ 的最短路径必定经过节点 $s$.因为 $m\in V-rsubtree(u,s)$,则节点 $m$ 到 $u$ 的最短路径必定不经过节点 $s$,得出矛盾.因此定理得证. □

为了实现定理 7,节点 $s$ 需要寻找属于集合 $neighbor(s)\cap(V-rsubtree(s))$中的节点,其中,$neighbor(s)$很容易判断,然而为了判断节点是否属于集合 $V-rsubtree(u,s)$,需要构造以 $u$ 为根的反向最短路径树 $rspt(u)$.但是在实际计算过程中,并没有必要构造完整的 $rspt(u)$,因为只要计算出节点 $u$ 的备份下一跳,算法就可以终止,因此可以降低算法的执行时间.当该节点存在多个备份下一跳时,算法尽量选择重路由路径代价最小的节点作为备份下一跳.只有当该节点无可用备份下一跳或者只有一个备份下一跳时,才有可能计算完整的 $rspt(u)$,其他情况只需要构造部分 $rspt(u)$即可.定理 7 给出了计算链路保护的方法:构造以节点 $u$ 为根的反向最短路径树 $rspt(u)$,当某个节点 $m$ 加入到 $rspt(u)$时,判断该节点是否属于集合 $neighbor(s)\cap(V-rsubtree(s))$:如果 $m$ 属于上述集合,则 $m\in backn(s,u)$;否则继续加入其他节点,直到找到节点 $u$ 的备份下一跳.下面通过定理 8 来说明利用定理 7 计算出的重路由路径具有最小代价,从而降低重路由路径拉伸度.

**定理 8**. 如果 $m\in backn(s,u)$,则不存在节点 $n\in backn(s,u)$使得 $cost(n,u)<cost(m,u)$.

证明:下面利用反证法来证明该定理.假设存在节点$n\in backn(s,u)$使得$cost(n,u)<cost(m,u)$,则可以得到节点$n$比节点 $m$ 先加入到以 $u$ 为根的最短路径树中.然而这不可能的,因为只要出现符合备份条件的节点加入到该树中,算法立即终止,因此假设不成立.定理得证. □

### 4.2 节点保护条件

在 $spt(s)$中,如果链路$(s,u)\in spt(s)$,当节点 $u$ 出现故障时,节点 $s$ 需要为 $child(s,u)$中的节点计算备份下一跳.因此为了保护某个节点,节点 $s$ 需要为该节点的所有孩子节点计算备份下一跳.

定理 9 给出了提供了如何为节点 $u$ 的孩子节点 $v$ 计算备份下一跳的方法:构造以节点 $v$ 为根的反向最短路径树 $rspt(v)$,当某个节点 $m$ 加入到 $rspt(v)$时,判断该节点是否属于集合 $neighbor(s)\cap(V-rsubtree(v,s)-rsubtree(v,u))$:如果 $m$ 属于上述集合,则 $m\in backn(s,v)$;否则继续加入其他节点,直到找到节点 $v$ 的备份下一跳.

**定理 9**. 如果$(s,u)\in sp(s,v)$,$(u,v)\in sp(s,v)$,$neighbor(s)\cap(V-rsubtree(v,s)-rsubtree(v,u))$中的所有节点都可以作为节点 $v$ 的备份下一跳,即:$neighbor(s)\cap(V-rsubtree(v,s)-rsubtree(v,u))\subset backn(s,v)$.

定理 9 的证明过程和定理 7 的证明过程类似,因此不再对其证明.

### 4.3 链路保护算法

算法 2 描述了如何为节点 $u$ 计算备份下一跳,算法需要构造以节点 $u$ 为根的反向最短路径树 $rspt(u)$.首先,将节点 $s$ 和 $subtree(s,u)$中所有节点标记为红色(算法 2 中的第 1 行、第 1 行);通过初始化操作,将节点 $u$ 加入到优先级队列 $Q$ 中(算法 2 中的第 3 行~第 9 行);构造树要经历一系列的迭代过程,在每一次迭戈中,从优先级队列中选取代价最小的节点 $y$(算法 2 中的第 11 行);如果其父亲节点的颜色是红色,则将该节点标记为红色(算法 2 中的第 13 行~第 15 行);如果该节点不是红色并且该节点是 $s$ 的邻居节点,则该节点即是节点 $u$ 的备份下一跳;否则,更新该节点的信息,并且将该节点加入到树中(算法 2 中的第 20 行~第 22 行);访问节点 $y$ 的所有邻居节点,更新这些邻居节点的信息(算法 2 中的第 24 行~第 31 行);

- 如果是针对节点保护问题,则只需要将链路保护算法中所有的变量 $u$ 改为 $v$ 即可.

**算法 2**. AsyLinkProtection.

Input: $SPT(s),u,G=(V,E)$;

Output: $backn(s,u)$.

```
1:  将 subtree(s,u)中所有节点标记为红色
2:  将节点 s 标记为红色
3:  For x∈V do
4:     cost(u,x)←∞
5:     u.visited←false
6:  EndFor;
7:  u.visited←true
8:  cost(u,u)←0
9:  Enqueue(Q,(u,u,0));
10: While Q is not empty do
11:    ⟨y,p,tc⟩←ExtractMin(Q)
12:    If y≠u then
13:       If parent(u,y)是红色 then
14:          将节点 y 标记为红色
15:       EndIf
16:       If y 不是红色 并且 y∈neighbor(s) then
17:          backn(s,u)←y
18:          return;
19:       EndIf
20:          y.visited←true
21:       parent(u,y)←p
22:          cost(u,y)←tc
23:    EndIf
24:    For q∈neighbor(y) do
25:       If q.visited←false then
26:          newdist←cost(y,u)+w(q,y)
27:             If newdist≤cost(q,u) then
28:             Enqueue(Q,(u,q,newdist))
29:          EndIf
30:       EndIf
```

31: **EndFor**

32: **EndWhile**

### 4.4 节点保护算法

根据定理 9 可知:为了实现节点保护算法,只需要将链路保护算法中所有的变量 $u$ 改为 $v$ 即可.节点保护算法用 AsyNodeProtection 表示.

## 5 算法讨论

从第 3 节和第 4 节的描述可知,本文提出了两种高效的 LFA 实现方法.

AsyLinkProtection 和 AsyNodeProtection 算法不需要考虑网络中链路权值是否对称,因此适用范围更加广泛.SynLinkProtection 和 SynNodeProtection 只适用于链路权值对称的网络.当网路出现单故障时,本文提出的算法只需要为特定的节点计算备份下一跳,根据定理 1 可知,其子树中所有节点的备份下一跳和该特定节点的备份下一跳是相同的.在某些情况下,当某个特定节点不存在备份下一跳时,利用本文提出的算法将导致该节点对应的子树中所有节点无法找到备份下一跳.因此,利用本文提出的算法可能会漏掉某些节点的备份下一跳.因此,为了提高故障保护率,在执行上述算法的过程中,如果某个节点没有备份下一跳,则将特定节点的计算范围扩大到该节点的下一跳.例如在图 1 中,为了保护链路$(s,u)$,节点 $s$ 只需为节点 $u$ 计算备份下一跳,因为 $u$ 对应的子树中所有节点的备份下一跳时相同的,如果节点 $u$ 不存在备份下一跳时,则为节点 $b$ 和 $c$ 计算备份下一跳.从下面的实验可以看出,该过程不会明显增加算法的计算开销,因此将特定节点的范围扩大到下一跳是提高故障保护率的一种有效解决方案.

## 6 实验及结果分析

### 6.1 实验方法

(1) 实验拓扑

为了全面准确评价算法的性能,采用 3 种类型的拓扑进行模拟实验,包括:真实拓扑 Abilene[43]、利用 Rocketfuel[44]测量的拓扑结构、利用 Brite[45]模拟软件生成的拓扑结构.

1) Abilene 是美国的教育和科研网络,其包含 11 个路由器和 14 条链路;
2) Rocketfuel 项目公布了大量的测量拓扑结构,我们选择其中的 6 个作为实验拓扑,具体参数见表 2;
3) 利用开源软件 Brite 生成拓扑结构,Brite 的具体参数见表 3.

**Table 2** Rocketfuel topology

**表 2** Rocketfuel 拓扑

| AS 号码 | AS 名称 | 结点数量 | 链路数量 |
|---|---|---|---|
| 1221 | Telstra | 108 | 153 |
| 1239 | Sprint | 315 | 972 |
| 1755 | Ebone | 87 | 162 |
| 3257 | Tiscali | 161 | 328 |
| 3967 | Exodus | 79 | 147 |
| 6461 | Abovenet | 128 | 372 |

**Table 3** Parameters for Brite topology

**表 3** Brite 生成拓扑结构的参数设置

| 模型 | 节点数量 | HS | LS |
|---|---|---|---|
| Waxman | 20~1000 | 1 000 | 100 |
| 链路节点比 | NodePlacement | 增长方式 | alpha |
| 2-40 | Random | 增量式 | 0.15 |
| beta | BWDist | BwMin-BwMax | 模式 |
| 0.2 | Constant | 10.0-1024.0 | 路由器 |

(2) 评价指标

由于本文提出的算法主要针对 IPFRR 中的 LFA 方法进行改进,因此为了评价本文算法的性能,在实验中将与 LFA 方法、TBFH 进行比较.评价指标包括计算开销、路径拉伸度和故障保护率.下面介绍详细的实验方法.

本文利用 C++语言实现了 LFA,TBFH 和本文提出的方案.在实验中,仅仅列出了对称网络中的实验数据,而非对称网络中的结果与之基本类似,因此没有详细列出.

### 6.2 计算开销

为了避免运行环境对算法性能的影响,本文采用相对计算时间来衡量不同算法的计算效率.相对计算时间可以定义为:相对计算时间=算法实际运行时间/构造最短路径树时间.从定义中可以看出,相对计算时间表示构造最短路径树的次数.下面通过模拟实验评价不同算法的相对计算时间.

首先,我们说明真实拓扑和 Rocketfuel 测量拓扑的计算结果.图 2 和图 3 分别描绘了不同算法在上述拓扑上链路保护和节点保护的相对计算时间.可以看出,本文提出的算法的执行效率明显优于 LFA 算法和 TBFH 算法.在链路保护中,SynLinkProtection 和 AsyLinkProtection 的计算开销基本接近.在节点保护中,虽然 AsyNodeProtection 的计算开销略大于 SynNodeProtection 的计算开销,但是明显优于 LFA 的性能,这是因为 SynNodeProtection 算法并不需要构造一棵完整的最短路径树,而 AsyNodeProtection 在某些情况下可能需要构造多棵完整的最短路径树.

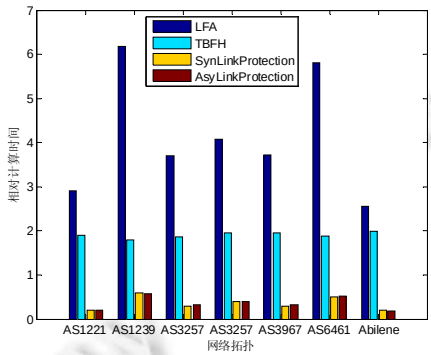


Fig.2 Computation overhead with link protection in real and measured topologies

图 2 真实和测量拓扑中链路保护计算开销

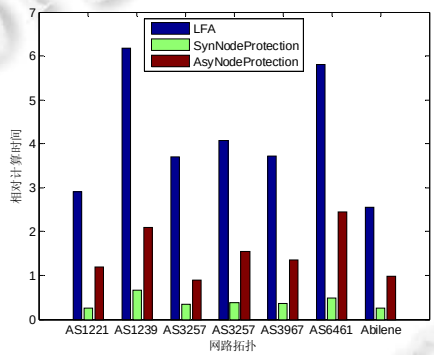


Fig.3 Computation overhead with node protection in real and measured topologies

图 3 真实和测量拓扑中节点保护计算开销

接着,介绍不同算法在 Brite 生成拓扑上的运行结果.图 4 和图 5 分别描述了当网络节点的平均度为 6,链路保护算法和节点保护算法的计算效率随着网络规模的变化规律.

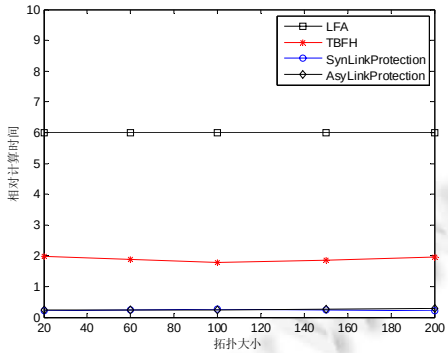


Fig.4 Computation overhead with link protection in generated topologies

图 4 Brite 生成拓扑中链路保护计算开销

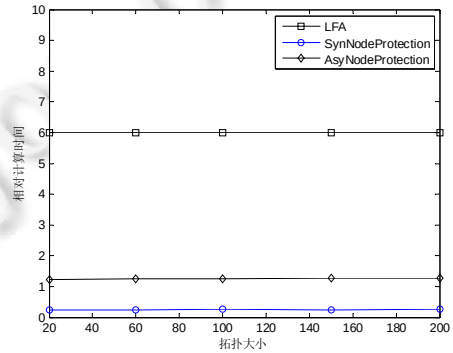


Fig.5 Computation overhead with node protection in generated topologies

图 5 Brite 生成拓扑中节点保护计算开销

图 6 和图 7 分别描述了当网络拓扑大小为 1 000,链路保护算法和节点保护算法的计算效率随着网络节点的平均度变化规律.

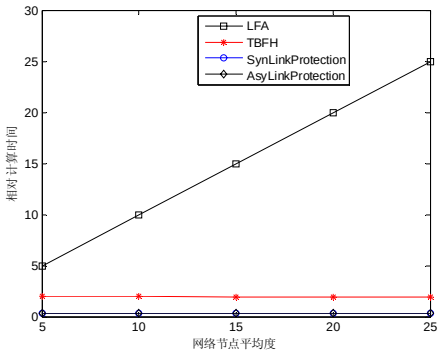


Fig.6 Computation overhead with link protection in generated topologies

图 6 Brite 生成拓扑中链路保护计算开销

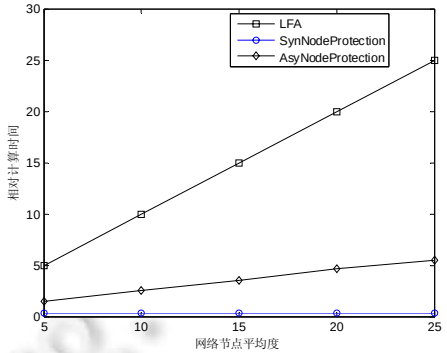


Fig.7 Computation overhead with node protection in generated topologies

图 7 Brite 生成拓扑中节点保护计算开销

从上述图可以得出:当网络节点的平均度确定后,所有算法的相对计算时间基本不受网络规模的影响.当网络规模确定后,LFA 算法随着网络节点平均度的增加而增加;本文提出的算法和 TBFH 算法基本不受该因素的影响,然而 TBFH 的执行效率明显低于本文提出的算法.

### 6.3 路径拉伸度

当网络出现故障时,利用路由保护算法计算出的重路由路径并不是针对新的拓扑结构计算的最短路径,因此,利用路由保护算法计算出的重路由路径的代价一定大于新拓扑对应的最短路径代价,必然引起路径的拉伸.因此,本节利用路径拉伸度来衡量重路由路径的优劣.路径拉伸度可以定义为:路径拉伸度=重路由路径的代价/最短路径代价.因此,当网络出现故障时,路由拉伸度越小,对应的重路由路径越接近最短路径,端到端延迟越小.本小节将评价当网络中发生单故障时(单链路或者单节点),不同算法对应的路径拉伸度.下面介绍实验方法,对于任意拓扑结构,随机选择一条链路断开,然后执行上述算法,计算不同算法对应的路径拉伸度.在实验中,选择50%的链路执行上述操作,最后取平均值.重复上述实验 100 次,最后得到实验结果.上面描述了链路保护的实验方法,节点保护的方法和上述方法类似,因此不再介绍.在实验比较中,对于 LFA 和 TBFH 方案,如果存在多个备份下一跳,则从中随机选择一个作为其备份下一跳.

首先,我们介绍不同算法在真实拓扑和 Rocketfuel 测量拓扑的实验结果.图 8 和图 9 分别描述了不同算法对应的链路保护和节点保护的路径拉伸度.

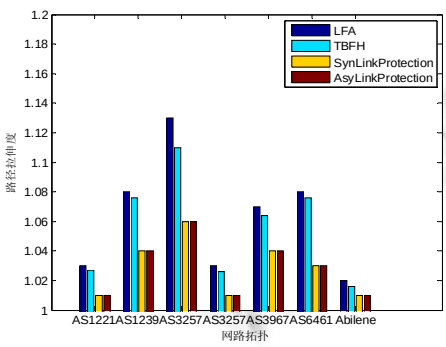


Fig.8 Path Stretch with link protection in real and measured topologies

图 8 真实和测量拓扑中链路保护路径拉伸度

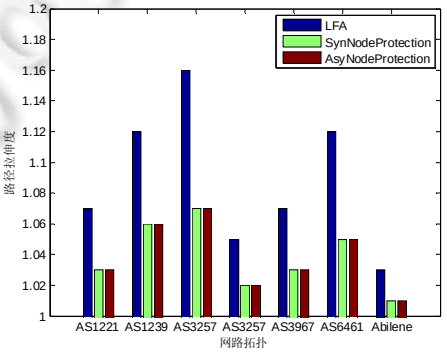


Fig.9 Path Stretch with node protection in real and measured topologies

图 9 真实和测量拓扑中节点保护路径拉伸度

从图中可以看出,LFA 和 TBFH 的路径拉伸度明显高于本文提出的算法.不论在链路保护还是在节点保护,本文提出的两种方案的路由拉伸度基本一致.这是因为我们提出方案选择代价最小的路径作为重路由路径,而 LFA 和 TBFH 随机选择其中一个作为重路由路径,从而导致其重路由路径拉伸度较大.

接着介绍不同算法在 Brite 生成拓扑上的运行结果.图 10 和图 11 分别描述了对应的链路保护和节点保护的路径拉伸度.从图中可以得出,算法在生成拓扑的运行的结果和在测量拓扑的运行结果基本一致.

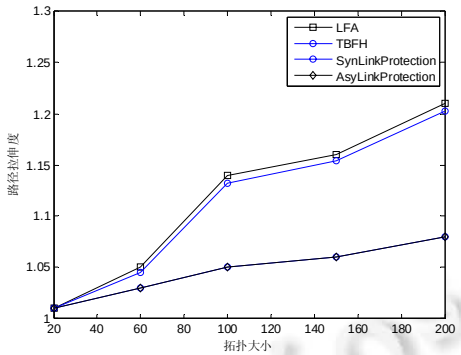


Fig.10 Path Stretch with link protection in generated topologies

图 10 Brite 生成拓扑中链路保护路径拉伸度

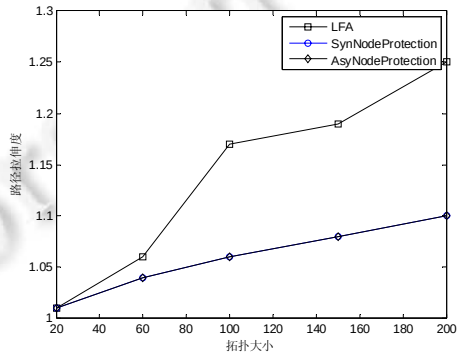


Fig.11 Path Stretch with node protection in generated topologies

图 11 Brite 生成拓扑中节点保护路径拉伸度

### 6.4 故障保护率

本节将用故障保护率来衡量不同算法应对故障的能力.故障保护率可以定义为:

$$p=\frac{\sum_{\forall s,d\in V}B(s,d)}{V*(V-1)}.$$

其中,

$$B(s,d)=\begin{cases}1, & backs(s,d)\neq\varnothing\\ 0, & backs(s,d)=\varnothing\end{cases}.$$

对于网络中任意节点$\forall s,d\in V$,如果 $s$ 到 $d$ 具有备份下一跳,则 $B(s,d)=1$;否则,$B(s,d)=0$.

图 12 和图 13 分别描述了不同算法在真实拓扑、Rocketfuel 测量拓扑对应的链路保护和节点保护的故障保护率.

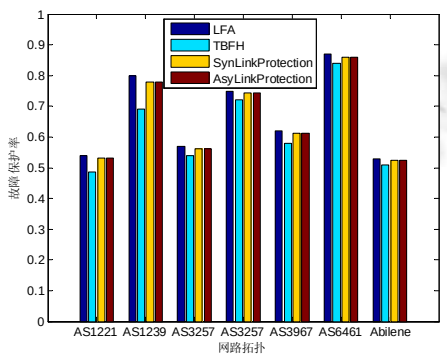


Fig.12 Protection rate with link protection in real and measured topologies

图 12 真实和测量拓扑中链路保护故障保护率

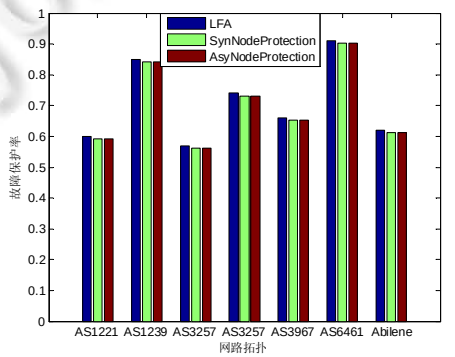


Fig.13 Protection rate with node protection in real and measured topologies

图 13 真实和测量拓扑中节点保护故障保护率

从图中可以看出,本文提出的算法和 LFA 算法的故障保护率基本一致.因此,与 LFA 比较,本文提出的算法不会降低路由可用性,TBFH 算法降低了路由可用性.

### 6.5 增量部署

本文提出的方案和互联网部署的域内路由协议是兼容的,因此可以在网络中增量部署该方案.增量部署方案可以描述为:给定具体的网络拓扑结和部署节点数量,选择合适的节点部署上述算法,从而使得故障保护率最高.因为网络中不同节点的重要程度是不相同的,因此实验中采用节点的介数来衡量节点的重要程度.下面使用贪心算法来解决该问题:首先,按照节点的介数对网络中所有节点进行降序排列;其次,每次从队列首部选择一个节点部署上述算法,直到不满足部署条件要求.

图 14 和图 15 分别描述了链路保护算法和节点保护算法在 Sprint 拓扑上部署节点数量和故障保护率之间的规律.从该图可以看出:随着部署节点数量的增加,故障保护率随之提高.当部署大约 40%左右的关键节点时,故障保护率已经得到明显提升.因此在实际中部署时,应该将不同的节点区分对待,优先部署重要节点.

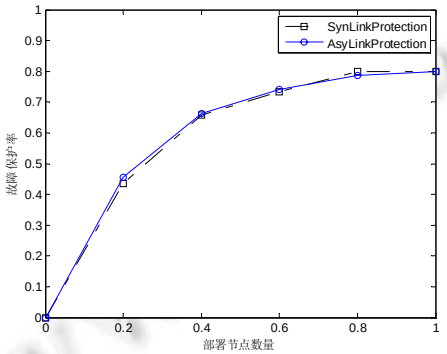


Fig.14 Deployment with link protection in Sprint topology

图 14 Sprint 拓扑中链路保护部署情况

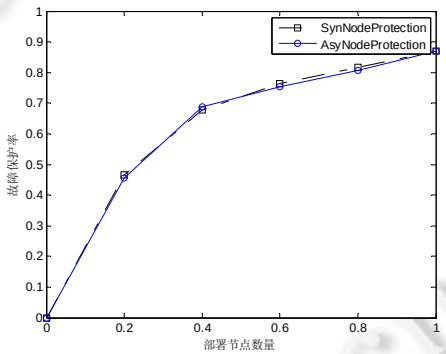


Fig.15 Deployment with node protection in Sprint topology

图 15 Sprint 拓扑中节点保护部署情况

## 7 总结与展望

针对目前互联网部署的 LFA 算法开销大的问题,本文设计了一种轻量级的基于逐跳方式的 IP 路由保护方案.理论和实验结果表明:与 LFA 算法相比较,本文提出的方案不仅计算复杂度低、路径拉伸度小,并且可以提供同样的故障保护率.然而,本文研究的对象是网络中单故障情形,因此下一步主要研究如何将本文的算法应用于并发故障的情形.

**References**:

[1] Internet users. http://www.internetworldstats.com/top20.htm

[2] Reaching 50 million users. http://visual.ly/reaching-50-million-users

[3] Chabarek J, Sommers J, Barford P, *et al*. Power awareness in network design and routing. In: Proc. of the IEEE Conf. on Computer Communications. IEEE INFOCOM, 2008. 457−465.

[4] Varshney U, Snow A, Mcgivern M, *et al*. Voice over IP. Communications of the ACM, 2002,45(1):89−96.

[5] Goode B. Voice over internet protocol (voip). Proc. of the IEEE, 2002,90(9):1495−1517.

[6] Drew P, Gallon C. Next-Generation voip network architecture. In: Proc. of the Multiservice Switching Forum. 2003. 1−19.

[7] Tapolcai J, Retvari G, Babarczi P, Berczi-Kovacsy ER, Kristofy P, Enyediz G. Scalable and efficient multipath routing: Complexity and algorithms. In: Proc. of the 2015 IEEE 23rd Int'l Conf. on Network Protocols (ICNP). IEEE, 2015. 376−385.

[8] Zheng JQ, Xu H, Zhu XJ, Chen GH, Geng YH. We've got you covered: Failure recovery with backup tunnels in traffic engineering. In: Proc. of the 2016 IEEE 24th Int'l Conf. on Network Protocols (ICNP). IEEE, 2016. 1−10.

[9] Markopoulou A, Iannaccone G, Bhattacharyya S, Chuah CN, Ganjali Y, Diot C. Characterization of failures in an operational IP backbone network. IEEE/ACM Trans. on Networking, 2008,16(4):749−762.

[10] Hou MJ, Wang D, Xu MW, Yang JH. Selective protection: A cost-efficient backup scheme for link state routing. In: Proc. of the IEEE Int'l Conf. on Distributed Computing Systems (ICDCS) 2009. 68−75.

[11] Francois P, Bonaventure O. Avoiding transient loops during the convergence of link-state routing protocols. IEEE/ACM Trans. on Networking, 2007,15(6):1280−1292.

[12] Yang B, Liu J, Shenker S, *et al*. Keep forwarding: Towards *k*-link failure resilient routing. In: Proc. of the IEEE Conf. on Computer Communications. IEEE INFOCOM, 2014. 1617−1625.

[13] Xu MW, Hou MJ, Wang D, Yang JH. An efficient critical protection scheme for intra-domain routing using link characteristics. Computer Networks, 2013,57(1):117−133.

[14] Kos A, Klepec B, Tomasic S. Challenges for voip technologies in corporate environments. In: Proc. of the ICN. 2004. 1−4.

[15] Pal S, Gadde R, Latchman HA. On the reliability of voice over ip (voip) telephony. In: Proc. of the SPRING 9th Int'l Conf. on Computing, Communications and Control Technologies. Orlando, 2011. 1−6.

[16] Xu A, Bi J, Zhang BB, Wang SH, Wu JP. Failure inference for shortening traffic detours. In: Proc. of the IEEE/ACM Int'l Symp. on Quality of Service (IWQoS). 2017. 1−10.

[17] Kwong KW, Gao L, Zhang ZL. On the feasibility and efficacy of protection routing in IP networks. IEEE/ACM Trans. on Networking, 2011,19(5):1543−1556.

[18] Peng Q, Walid A, Low SH. Multipath TCP algorithms: Theory and design. In: Proc. of the ACM SIGMETRICS Int'l Conf. on Measurement and Modeling of Computer Systems. 2013. 305−316.

[19] Gran EG, Dreibholz T, Kvalbein A. NorNet core—A multihomed research testbed. Computer Networks, 2014,61(C):75−87.

[20] Atlas A, Kebler R, Konstantynowicz M, *et al*. An architecture for IP/ LDP fast-reroute using maximally redundant trees. Standards Track, 2015. 1−41.

[21] Enyedi G, Csaszar A, Atlas A, Bowers C, Gopalan A. Algorithms for computing maximally redundant trees for IP/LDP fast-reroute. Informational, 2013. 1−56.

[22] Lee S, Yu Y, Nelakuditi S, Zhang ZL, Chuah CN. Proactive vs reactive approaches to failure resilient routing. In: Proc. of the IEEE INFOCOM. Hong Kong, 2004. 1−11.

[23] Cho S, Elhourani T, Ramasubramanian S. Independent directed acyclic graphs for resilient multipath routing. IEEE/ACM Trans. on Networking (TON), 2012,20(1):153−162.

[24] Enyedi G, Rétvári G, Szilágyi P, Császár A. IP fast ReRoute: Lightweight not-via without additional addresses. In: Proc. of the INFOCOM. 2009. 2771−2775.

[25] Xu M, Yang Y, Li Q. Selecting shorter alternate paths for tunnel-based IP fast ReRoute in linear time. Computer Networks, 2012, 56(2):845−857.

[26] Banerjee G, Sidhu D. Comparative analysis of path computation techniques for MPLS traffic engineering. Computer Networks, 2002,40(1):149−165.

[27] Atlas AK, Zinin A. Basic specification for IP fast reroute: Loop-free alternates. RFC 5286, 2008. 1−32.

[28] Shaikh A, Greenberg A. Experience in black-box OSPF measurement. In: Proc. of the ACM SIGCOMM Workshop on Internet Measurement. ACM Press, 2001. 113−125.

[29] Alaettinoglu C, Jacobson V, Yu H. Towards milli-second IGP convergence. IETF Draft, 2000. 1−8.

[30] Francois P, Filsfils C, Evans J, Bonaventure O. Achieving sub-second IGP convergence in large IP networks. Computer Communication Review, 2005,35(3):35−44.

[31] Mérindol P, Francois P, Bonaventure O, Cateloin BS, Pansiot JJ. An efficient algorithm to enable path diversity in link state routing networks. Computer Networks, 2011,55(5):1132−1149.

[32] Gjoka M, Ram V, Yang X. Evaluation of IP fast reroute proposals. In: Proc. of the Int'l Conf. on Communication Systems Software and MIDDLEWARE. IEEE, 2007. 1−8.

[33] Rétvári G, Csikor L,Tapolcai J, *et al*. Optimizing IGP link costs for improving IP-level resilience. In: Proc. of the Design of Reliable Communication Networks (DRCN). 2011. 62−69.

[34] Rétvári G, Tapolcai J, Enyedi G, *et al*. Ip fast reroute: Loop free alternates revisited. In: Proc. of the INFOCOM. 2011. 2948−2956.

[35] Narvaez P, Siu K, Tzeng H. New dynamic SPT algorithm based on a ball-and-string model. IEEE/ACM Trans. on Networking (TON), 2001,9(6):706−718.

[36] Francois P, Bonaventure O. Avoiding transient loops during the convergence of link-state routing protocols. IEEE/ACM Trans. on Networking, 2007,15(6):1280−1292.

[37] Moy J. Ospf version 2. RFC 2328, 1998. 1−244.

[38] Sridharan A, Guerin R, Diot C. Achieving near-optimal traffic engineering solutions for current ospf/is-is networks. IEEE/ACM Trans. on Networking (TON), 2005,13(2):234−247.

[39] Yang X, Wetherall D. Source selectable path diversity via routing deflections. In: Proc. of the SIGCOMM. 2006. 159−170.

[40] Kvalbein A, Hansen AF, Gjessing S, Cicic T, Gjessing S, Lysne O. Fast IP network recovery using multiple routing configurations. In: Proc. of the IEEE Int'l Conf. on Computer Communications. 2007. 1−11.

[41] Lakshminarayanan K, Caesar M, Rangan M, Anderson T, Shenker S, Stoica I. Achieving convergence-free routing using failure-carrying packets. ACM SIGCOMM Computer Communication Review, 2007,37(4):241−252.

[42] Sommers J, Barford P, Eriksson B. On the prevalence and characteristics of MPLS deployments in the open Internet. In: Proc. of the 2011 ACM SIGCOMM Conf. on Internet Measurement Conf. 2011. 445−462.

[43] https://www.internet2.edu/products-services/advanced-networking

[44] Spring N, Mahajan R, Wetherall D, Anderson T. Measuring ISP topologies with rocketfuel. IEEE/ACM Trans. on Networking, 2004,12(1):2−16.

[45] http://www.cs.bu.edu/brite/

**耿海军**(1983－),男,山西灵石人,博士,讲师, CCF 专业会员,主要研究领域为软件定义网络,路由算法,网络体系结构.

**施新刚**(1980－),男,博士,高级工程师,主要研究领域为路由协议,网络测量.

**王之梁**(1978－),男,博士,副研究员,主要研究领域为下一代互联网,路由算法.

**尹霞**(1972－),女,博士,教授,博士生导师,CCF 高级会员,主要研究领域为下一代互联网,协议测试.

**尹少平**(1965－),男,硕士,副教授,CCF 专业会员,主要研究领域为软件定义网络,路由算法,网络体系结构.