计算机科学
COMPUTER SCIENCE

基于软件定义网络的高故障保护率的路由保护方案

耿海军, 王威, 张晗, 王玲

引用本文

耿海军, 王威, 张晗, 王玲. 基于软件定义网络的高故障保护率的路由保护方案[J]. 计算机科学, 2023, 50(9): 337-346.

GENG Haijun, WANG Wei, ZHANG Han, WANG Ling. Routing Protection Scheme with High Failure Protection Ratio Based on Software-defined Network [J]. Computer Science, 2023, 50(9): 337-346.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.220900220

基于软件定义网络的高故障保护率的路由保护方案

耿海军[1,2,3]　王　威[1,3]　张　晗[4]　王　玲[2]
1 山西大学计算机与信息技术学院　太原 030006
2 山西大学自动化与软件学院　太原 030006
3 山西大学大数据科学与产业研究院　太原 030006
4 清华大学计算机科学与技术系　北京 100084

摘　要　软件定义网络(Software Defined Network,SDN)以其强大的可编程性和集中控制的优势得到了学术界的广泛关注。现有的SDN设备在执行报文转发时仍然使用最短路径协议,当最短路径中的结点发生故障时,网络仍然需要重新收敛,在此期间报文可能会被丢弃,进而无法传递至目的结点,给实时性应用的流畅性造成了冲击,影响用户体验。学术界普遍采用路由保护的方案来应对网络故障,现有的路由保护方案存在以下两个方面的问题:(1)故障保护率低;(2)当网络出现故障时,备份路径可能会出现路由环路。为了解决上述两个问题,首先提出了备份下一跳计算规则;然后基于此规则设计了一种软件定义网络下的高故障保护率的路由保护算法(Routing Protection Algorithm with High Failure Protection Ratio,RPAHFPR),该算法融合了路径生成算法(Path Generation Algorithm,PGA)、旁支优先算法(Side Branch First Algorithm,SBF)和环路规避算法(Loop Avoidance Algorithm,LAA),可以同时解决已有路由保护方法面临的故障保护率低和路由环路问题;最后在大量的真实网络拓扑和模拟网络拓扑中验证了RPAHFPR方案的性能。与经典的NPC和U-TURN相比,RPAHFPR的故障保护率分别提高了20.85%和11.88%,并且在86.3%的拓扑中可以达到100%的故障保护率,在所有拓扑中可以达到99%以上的故障保护率。RPAHFPR的路径拉伸度基本接近1,不会引入过多的时间延迟。

关键词:软件定义网络;路由保护算法;反向最短路径树;LFA规则;备份路径;网络单故障
中图法分类号　TP311

Routing Protection Scheme with High Failure Protection Ratio Based on Software-defined Network

GENG Haijun[1,2,3],WANG Wei[1,3],ZHANG Han[4] and WANG Ling[2]
1 School of Computer and Information Technology,Shanxi University,Taiyuan 030006,China
2 School of Automation and Software Engineering,Shanxi University,Taiyuan 030006,China
3 Institute of Big Data Science and Industry,Shanxi University,Taiyuan 030006,China
4 Department of Computer Science Technology,Tsinghua University,Beijing 100084,China

Abstract　SDN has attracted extensive attention from academia for its advantages of strong programmability and centralized control. Existing SDN devices still use the shortest path protocol when performing packet forwarding. When a node in the shortest path fails,the network re-convergence is still required. During this period,packets may be discarded and thus cannot be delivered to the destination node,which has an impact on the flow of real-time applications and affects the user experience. The academia generally adopts the routing protection schemes to deal with network failures. The existing routing protection schemes have the following two problems:(1) the failure protection ratio is low;(2) when the network fails,the backup path may have routing loops. In order to solve the above two problems,a backup next hop calculation rule is proposed. Then,based on this rule,a routing protection algorithm with high hailure protection ratio(RPAHFPR) is designed,which combines the path generation algorithm (PGA),side branch first algorithm(SBF) and loop avoidance algorithm(LAA). It can simultaneously solve the low failure protec-

到稿日期:2022-09-23　返修日期:2023-03-05
基金项目:山西省应用基础研究计划(20210302123444);山西省高等学校科技创新项目(2022L002);中国高校产学研创新基金(2021FNA02009);国家自然科学基金(61702315);山西省重点研发计划(201903D421003,202202020101004);国家高技术研究发展计划(863)(2018YFB1800401)
This work was supported by the Shanxi Applied Basic Research Program(20210302123444),Scientific and Technological Innovation Programs of Higher Education Institutions in Shanxi(2022L002),Industry-University-Research Innovation Fund of Chinese Universities(2021FNA02009),National Natural Science Foundation of China(61702315),Key R&D Program of Shanxi Province,China(201903D421003,202202020101004) and National High Technology Research and Development Program of China(2018YFB1800401).
通信作者:耿海军(ghj123025449@163.com)

tion rate and routing loop problems faced by existing routing protection methods. Finally, the performance of RPAHFPR scheme is verified in a large number of real network topologies and simulated network topologies. Compared with the classic NPC and U-TURN, the failure protection rate of RPAHFPR is increased by 20.85% and 11.88% respectively, and it can achieve 100% failure protection rate in 86.3% topology, and more than 99% failure protection rate in all topology. The path stretching degree of RPAHFPR is basically close to 1, without introducing too much time delay.

Keywords Software-defined network, Route protection algorithm, Reverse shortest path tree, LFA rule, Backup path, Network single failure

1 引言

互联网是全球最重要的基础通信媒介之一，越来越多的应用程序都部署在互联网中。随着人工智能的兴起，股票交易、VoIP、在线视频等实时性应用[1-3]被广泛应用于社会生活中。与邮件传输等传统应用相比，这类应用对网络时速及流畅性更为敏感，对网络的“自愈能力”提出了更高的要求[4]。随着网络规模的不断扩大，网络故障的出现不可避免[5-6]。现行的解决方案是采用自适应动态路由协议，如 RIP，OSPF，IS-IS 等。这类协议都是通过路由收敛来解决故障，但是 RIP 的收敛时间为 100s 的量级，且适用于小型网络环境。OSPF 和 IS-IS 也需要十几秒甚至更多时间来完成收敛，还会伴随路由环路或路由黑洞的问题[7-9]。因此现行的路由协议已经无法满足用户需求，如何提高网络应对故障的能力已成为一个亟需解决的问题。

软件定义网络(SDN)是一种新型的网络体系结构[10]，其通过对传统网络设备中的控制平面和转发平面进行解耦，实现集中式控制。与传统网络相比，SDN 具有诸多优势，比如它能够对数据转发进行全局控制，简化设备运维，提高转发速率；此外 SDN 具有高度的可编程性和灵活性，控制平面通过北向接口和南向接口协议分别与应用平面和下层转发平面实现交互，其强大的可编程性和集中控制特点为大规模网络的路由保护提供了新的思路[11]。基于此，本文主要研究一种 SDN 环境中的路由保护方案，该方案应具有较高的故障保护率，以使大规模 SDN 网络能够应对绝大多数的网络故障。

本文的主要贡献如下：

1)现有的路由保护方案主要集中于传统的网络体系结构，随着 SDN 技术的不断成熟，SDN 的广泛部署是未来发展的必然趋势，因此在 SDN 环境中研究路由保护方案具有重要的现实意义。

2)提出了具有创新性的路由保护算法，并与其他单故障保护算法进行了实验对比，结果表明本文算法的故障保护率能够达到 99%以上，远高于同类单故障保护算法且不会带来庞大的路径代价消耗。

2 国内外研究现状

关于网络故障的大量研究证明网络故障出现频繁且不可避免[12]。常见的解决路由故障的方法有两类，分别是路由恢复方案和路由保护策略[13]。路由恢复方案属于被动保护方案，其通常是在网络故障发生后依靠修改路由协议的参数或链路的权值来提升网络的收敛速度。但网络的故障不可预知，因此该方法并不具有普适性，同时还会造成网络性能不稳定。针对此问题，路由保护策略在设计路由时预先计算出可能发生的故障结点以及对应的备份路径。当故障发生时，该策略可以通过备份路径巧妙地避开故障结点进而到达目的结点。最初的路由保护策略通过等价多路由机制(Equal Cost Multi-path，ECMP)来实现，ECMP 允许在源和目的之间的每个结点的多个传出链路上分发数据包，一旦路由器检测到故障，它会将具有相同成本的剩余链路上的数据包转发到目的地[14]。但实际网络中存在多条最短路径的情况非常有限，因此 ECMP 并不能够很好地提高网络的自愈性。针对这一问题，互联网工程任务组(IETF) 起草了一个名为 IPFRR[15-17]的框架，其基本思想是，当结点检测到与自身直接相连的链路出现故障时，它可以立即切换到专门为该故障计算的备份路径。在此基础上，IETF 又建立了无环备选项(Loop-Free Alternate，LFA)。LFA 是选择合适的备份下一跳的基本准则[18]，包括 LFC(Loop Free Condition)，DC(Downstream Condition)和 NPC(Node Protection Condition)。LFA 因其部署简单且无环路的优点而成为目前最受欢迎的路由保护方案，也是目前路由器供应商唯一提供的快速重路由方案。但是 LFA 也有其自身的局限性，LFA 的故障保护率较低，且保护性能很大程度上会受到网络拓扑的影响，无法保证能够应对网络中绝对大多数的链路或结点故障。针对这一问题，Csikor 等[19-20]通过优化链接成本来增加受保护结点的数量。rLFA[21-23]通过最短路径隧道将数据包重新路由到远程结点以增加受保护目的地的数量。它们不需要额外的转发条目，但仍不能保护所有目的地。LFA 和 rLFA 的性能都可以通过向网络添加链接来增强[24]。在文献[25]中，Tavernier 等提出了一个基于探针的 LFA 的自配置扩展，在其他结点中安装替代跃点以防止重新路由循环。与拓扑无关的 LFA(TI-LFA)[26]利用分段路由(Segment Routing，SR)[27]来防止故障。文献[28]在 LFC 规则的基础上，提出了 U-TURN 算法来提高故障保护率。U-TURN 将受故障影响的数据流量转发给存有 LFC 备份下一跳的邻居结点，具有比 LFA 更高的故障保护率，但仍无法应对绝大多数的单结点故障。针对 LFA 的缺点，文献[29]提出了 Not-Via 方案，该方案虽然能够解决网络中所有的单故障问题，但该算法的复杂度较高，无法在实际中部署。

文献[30]研究了如何在 SDN 中部署 ECMP 方法，仿真结果表明在 SDN 环境中实现的 ECMP 性能更优。文献[31]通过在 SDN 中部署 LFA 来增强 LFA 的故障保护能力。文献[32]提出了一种在软件定义网络(SDN)中具有流聚合的本地快速重路由(Local Fast Reroute，LFR)算法。在 LFR 中，如果检测到链路故障，所有受此故障影响的流量将聚合为

一个新的“大”流，然后由 SDN 控制器为聚合流动态部署本地重路由路径。实验结果表明 LFR 能够在有效减少 SDN 控制器和交换机之间的流数量的同时实现路由恢复。文献[33]针对 SDN 快速重路由中恢复路径计算的性能瓶颈问题，提出了一种在大规模 SDN 网络中快速索引最短恢复路径的算法，实验表明该算法能够显著降低恢复路径的代价。文献[34]研究了 SDN 中数据流量主动恢复算法，能够在基于 openflow 的网络中修复链路或者结点故障，该算法使用 VLAN 标记将备用路径设置的流存储减少了 99%，并成功地实现了 3～4 ms 左右的故障恢复。针对软件定义的广域网络故障恢复所需时间较长的问题，文献[35]开发了一种新的故障恢复系统 Sentinel。Sentinel 预计算并安装备份隧道，以加速故障恢复。当链路发生故障时，交换机可以在本地将流量重定向到备份隧道，并立即在数据平面上恢复，从而大大减少了瞬态拥塞。实验结果表明，Sentinel 可以有效地减少流量损失，提高容量利用率。文献[36]对混合 SDN 网络中的单链路故障问题进行了研究，并提出了一种启发式算法 SLFRPHSDN 求解该问题，仿真实验结果证明该算法只需要在传统网络中将少部分结点部署为 SDN 结点，就可以应对所有的单链路故障，但并未解决混合 SDN 中的单结点故障。针对此问题，文献[37]将在传统网络中部署 SDN 结点的问题抽象为整数规划问题，并提出启发式算法加以解决。经实验验证，该算法只需在传统网络中部署少量的 SDN 结点即可应对绝大部分单结点故障。

然而，已有的路由保护方案或者结构复杂且无法在实际中部署，或者需要管理员手动配置，还有一部分方案容易部署，但无法实现较高的故障保护率且性能不稳定。利用 SDN 集中控制和可编程性的优势，许多学者对传统的路由保护算法进行了优化，但都集中于缩短时间或减少流量消耗、避免拥塞方面的改进，故障保护率并没有得到较大的提高。然而 SDN 设备仍然通过最短路径转发报文，当网络结点发生故障时，仍然需要重新收敛，在此期间报文可能会被丢弃。因此本文主要研究如何在 SDN 环境下提升网络的自愈性，以使本文所提算法能够应对绝大多数的网络故障。

3 网络模型和问题描述

3.1 网络模型

本节首先定义一个网络模型，并通过该网络模型来描述要解决的关键问题以及重要定理。表 1 列出了文中所用到的部分符号，以便于读者阅读。

表 1 符号

Table 1 Notations

符号	含义
$G=(V,E)$	无向连通图
$nei(v)$	结点 v 的所有邻居结点的集合
$rspt(v)$	以结点 v 为根的反向最短路径树
$level(v,x)$	$rspt(v)$ 中结点 x 的层次
$father(v,x)$	$rspt(v)$ 中结点 x 的父亲结点
$ance(v,x)$	$rspt(v)$ 中结点 x 的所有祖先结点的集合
$subroot(v,x)$	$rspt(v)$ 中结点 x 的子树根结点
$bn(x,y)$	结点 x 到结点 y 的备份下一跳的集合
$besthop(x,y)$	结点 x 到结点 y 的最优下一跳
$sp(x,y)$	结点 x 到结点 y 的最短路径
$bp(x,y)$	结点 x 到结点 y 的备份转发路径

本文将网络描述成无向连通图 $G=(V,E)$，其中 V 为此网络中路由器(结点)的集合，E 为此网络中链路(边)的集合。

定义 1 对于任意的源-目的结点对(s,d)，当结点 s 到结点 d 的最优下一跳 $besthop(s,d)$ 发生故障时，若存在一个 s 的邻居结点 t 使得 $besthop(s,d)\notin bp(s,d)$ 成立，则称该结点对被保护，其备份下一跳为 t 。

定义 2 对于任意的源-目的结点对(s,d)，若结点 s 存在一个邻居结点 n，使得 $subroot(n)\neq subroot(s)$，则称邻居结点 n 为结点对(s,d)的旁支备份结点。

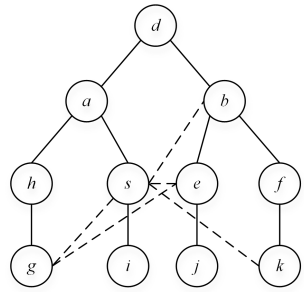

图 1 以目的结点 d 为根的反向最短路径树 $rspt(d)$

Fig. 1 Reverse shortest path tree $rspt(d)$ rooted at destination node d

下面将通过一个简单的例子来解释上述网络模型和定义。图 1 是一棵以目的结点 d 为根的反向最短路径树 $rspt(d)$，通过虚线连接的两个结点代表其在实际网络拓扑中是邻居结点。结点 s 在树中的层次为 3 可以表示为 $level(d,s)=3$，结点 s 在 $rspt(d)$ 的父亲结点可以表示为 $father(d,s)=a$，同时结点 a 是结点 s 到结点 d 的最优下一跳，即 $father(d,s)=besthop(s,d)=a$。结点 s 在 $rspt(d)$ 的祖先结点可以表示为 $ance(d,s)=\{a,d\}$。假设 s 在网络拓扑 G 中的邻居结点 $nei(s)=\{a,i,e,b,k,g\}$，结点 s 到结点 d 的默认转发路径为 $sp(s,d)=(s,a,d)$，当结点 a 发生故障时，结点 s 到结点 d 的备份下一跳为 $bn(s,d)=k$，备份路径为 $bp(s,d)=(s,k,f,b,d)$，备份路径中不存在故障结点 a，因此可以称结点对(s,d)被保护了。结点 s 的子树根结点为 $subroot(d,s)=a$，结点 k 的子树根结点为 $subroot(d,k)=b$，因此结点 s 和结点 k 分布在不同子树上，即结点 k 为结点 s 的旁支备份结点。关于使用旁支备份结点的作用将会在定理 4 中详细陈述。

3.2 问题描述

大数据时代下，人们对于众多实时应用的使用流畅性提出了更加严苛的要求。SDN 以其强大的可编程性和集中控制的优势得到了学术界的广泛关注，然而现有的 SDN 设备在执行报文转发时仍然使用最短路径协议，当最短路径中的结点发生故障时，网络仍然需要重新收敛，在此期间报文可能会被丢弃，进而无法传递至目的结点，给实时性应用的流畅性造成了冲击，影响用户体验。因此本文研究如何在 SDN 网络中实现一种路由保护算法，该算法能够解决路由中绝大多数的单故障问题。本文需要解决的关键问题可以描述为：对于给定网络拓扑 $G=(V,E)$，如何设计一种路由保护算法使得拓扑中尽可能多的源-目的结点对都被保护起来。该算法需要同时满足以下两个要求：

1)算法的故障保护率接近100%。

2)算法所带来的路径代价基本接近默认OSPF的路径代价。

4 备份下一跳计算规则

4.1 备份下一跳计算规则

为了高效地计算所有源-目的结点对的备份下一跳,本节首先介绍备份下一跳计算规则,然后从理论上证明其正确性。已知目的结点 d 和以 d 为根的反向最短路径树 $rspt(d)$,对于网络中的任意结点 $s(s\neq d)$,计算结点对(s,d)的备份下一跳结点的规则如下:

规则1 如果 $level(d,nei(s))=level(d,s)$和 $father(d,nei(s))\neq father(d,s)$同时成立,则 $bn(s,d)=bn(s,d)\cup nei(s)$。

规则2 如果 $level(d,nei(s))<level(d,s)$,则 $bn(s,d)=bn(s,d)\cup nei(s)$。

规则3 如果 $level(d,nei(s))>level(d,s)$且 $father(d,s)\notin ance(d,nei(s))$,则 $bn(s,d)=bn(s,d)\cup nei(s)$。

规则4 如果 $subroot(nei(s))\neq subroot(s)$,则 $bn(s,d)=bn(s,d)\cup nei(s)$。

下面通过定理来证明备份下一跳计算规则的正确性。

定理1 已知目的结点 d 和以 d 为根的反向最短路径树 $rspt(d)$,对于网络中的任意结点 $s(s\neq di)$,计算结点对(s,d)的备份下一跳结点。若 $level(d,nei(s))=level(d,s)$和 $father(d,nei(s))\neq father(d,s)$同时成立,则 $bn(s,d)=bn(s,d)\cup nei(s)$。

证明:利用反证法来证明其正确性。在 $rspt(d)$中,对于结点对(s,d),假设 $level(d.nei(s))=level(d,s)$和 $father(d,nei(s))=father(d,s)$同时成立,对于$(s,d)$,其最优下一跳为 $father(d,s)$,即 $father(d,s)$为故障结点,由 $father(d,nei(s))=father(d,s)$,可得 $nei(s)$的最优下一跳 $father(d,nei(s))$也发生了故障,因此 $nei(s)$无法作为(s,d)的备份结点,这与 $bn(s,d)=nei(s)$矛盾,假设不成立,证毕。

定理2 已知目的结点 d 和以 d 为根的反向最短路径树 $rspt(d)$,对于网络中的任意结点 $s(s\neq d)$,计算结点对(s,d)的备份下一跳结点。如果 $level(d,nei(s))<level(d,s)$,则 $bn(s,d)=bn(s,d)\cup nei(s)$。

证明:采用反证法来证明其正确性。在 $rspt(d)$中,对于结点对(s,d),假设 $level(d,nei(s))<level(d,s)$,$bn(s,d)\neq nei(s)$。事实上,当 $father(d,s)$发生故障时,结点 s 可以将报文转发给其邻居结点 $nei(s)$,这与 $bn(s,d)\neq nei(s)$矛盾,故假设不成立,证毕。

定理3 已知目的结点 d 和以 d 为根的反向最短路径树 $rspt(d)$,对于网络中的任意结点 $s(s\neq d)$,计算结点对(s,d)的备份下一跳结点。如果 $level(d,nei(s))>level(d,s)$且 $father(d,s)\notin ance(d,nei(s))$,则 $bn(s,d)=bn(s,d)\cup nei(s)$。

证明:采用反证法证明其正确性。在 $rspt(d)$中,对于结点对(s,d),假设 $level(d,nei(s))>level(d,s)$和 $father(d,s)\in ance(d,nei(s))$同时成立,$bn(s,d)=bn(s,d)\cup nei(s)$。当 $father(d,s)$发生故障时,结点 s 将报文转发给结点 $nei(s)$,由于 $father(d,s)\in ance(d,nei(s))$,因此转发过程仍会经过故障结点,与 $bn(s,d)=bn(s,d)\cup nei(s)$矛盾,假设不成立,证毕。

定理4 已知目的结点 d 和以 d 为根的反向最短路径树 $rspt(d)$,对于网络中的任意结点 $s(s\neq d)$,计算结点对(s,d)的备份下一跳结点。如果 $subroot(nei(s))\neq subroot(s)$,则 $bn(s,d)=bn(s,d)\cup nei(s)$。

证明:根据定义2,$nei(s)$为 s 的旁支备份结点,即 $nei(s)$与 s 不在同一棵子树中,但 s 的最优下一跳结点一定与结点 s 在同一棵子树上。因此当结点 s 的最优下一跳发生故障时,将报文发送给 $nei(s)$一定不会在转发过程中再次遇到故障结点,从而顺利到达目的结点。

4.2 备份下一跳计算实例

下面将通过举例进一步解释上述定理的含义。在图1所示的以目的结点 d 为根的反向最短路径树 $rspt(d)$中,s 在网络拓扑 G 中的6个邻居结点为 $nei(s)=\{a,i,e,b,k,g\}$,故障结点为 a,现在要计算(s,d)的备份下一跳结点。由定理1可知,结点 s 和结点 e 在树中的层次都为3且结点 s 和结点 e 的父亲结点不同,因此结点 e 可作为(s,d)的备份结点,备份路径为 $bp(s,d)=(s,e,b,d)$。根据定理2,结点 b 的层次高于结点 s 的层次,因此结点 b 可作为(s,d)的备份结点,备份路径为 $bp(d,c)=(s,b,d)$。根据定理3,结点 k 是结点 s 的下层结点且结点 s 的父亲结点 a(故障结点)不存在于结点 k 的祖先结点中,因此结点 k 可作为(s,d)的备份结点。但是结点 g 不能作为(s,d)的备份下一跳结点,这是因为故障结点 a 是结点 g 的祖先结点,当结点 g 收到报文后按照其最优路径(g,h,a,d)转发仍然会遇到故障结点 a,无法顺利将报文传递给目的结点。下面假设这样一种情况:h 是源结点,它要将报文传递给目的结点 d,(h,d)唯一的备份结点为 g,(g,d)有 s,e 两个可选备份结点。当 h 的最优下一跳结点 a 发生故障时,h 会将报文发送给其备份下一跳 g,g 收到最优下一跳发来的报文时会将报文发送给(g,d)的备份结点,此时如果将报文发送至结点 s,则其备份路径为(h,g,s,a,d)会再次遇到故障结点 a,无法将报文传递至目的结点 d,因此就形成了报文始终在故障结点所在的子树中转发而无法到达目的结点的情况。但若结点 g 将报文发送至其旁支备份结点 e,e 收到报文后按照其最优下一跳正常转发,得到的备份路径为(h,g,e,b,d)完全不会遇到故障结点 a,因此报文可以顺利到达目的结点。由此可见,旁支备份结点可以避免子树根结点发生故障所带来的路由中断问题,路由保护效果更加稳定,因此我们在选择备份下一跳结点时会优先选择旁支备份结点。关于 g 收到最优下一跳发来的报文时将其发送至备份结点的原因会在算法3路径生成算法(PGA)中阐述。

5 算法实现

5.1 RPAHFPR算法

图2所示为RPAHFPR算法的基本流程,整个过程分两轮进行,第一轮是根据定理1—定理4的规则,计算得到结点

对的备份下一跳，第二轮是针对定理1－定理4无法保护的结点对，遍历未保护结点对的邻居结点，根据这些邻居结点所存储的备份下一跳结点，动态地计算备份路径。算法1给出了RPAHFPR的具体执行过程：对于网络 G，以目的结点 d 为根构造反向最短路径树，若 G 中有 v 个结点就会产生 v 棵不同根结点的反向最短路径树。在每一棵反向最短路径树中，计算所有结点到根结点的备份下一跳，M 是用于存放备份结点的集合，bn 是一个字典，字典的键为结点对，值为该结点对从 M 中最终选择的最优备份结点，初始置为空（第1－2行）。对网络 G 的每一组结点对 (s,d)，遍历 s 的所有邻居结点，将符合定理1－定理3的结点加入 M 中（第3－9行）。利用SBF算法判断 M 中是否有结点符合定理4，如果存在则更新 M，使得 M 只保留符合定理4的结点，即优先保留不与计算结点在同一分支的备份结点（第10行）。因此 M 中存放着能够作为当前结点对的备份结点集合，利用路径生成算法（PGA）计算每个备份结点下生成的备份路径，并将备份路径代价最小的备份下一跳存入 bn 中，为避免计算出的备份路径出现环路，在PGA算法中设置了环路规避算法（LAA）（第11－12行）。当执行完上述循环后，第一轮计算完成（第13－14行）。针对还没有备份结点的结点对，进行第二轮计算。遍历无备份结点的结点对，$sign(s,d)$ 函数对 (s,d) 进行标记，如果 s 结点的邻居结点存在备份结点，则标记为1，否则标记为0。$degree(s)$ 是计算 s 结点的度，R 为标记字典，用来存放每一对结点的标记和度。我们根据 s 结点的度对 R 中的结点对降序排序（第15－17行），度越大的结点，连通性越好。当其所在的结点对找到了备份结点，其对整个网络性能的提升就越大，因此我们优先考虑度越高的结点对，尽可能地避免计算顺序导致的无备份结点的情况发生。接下来进行一个循环，依次遍历排序后的 R 字典，R 中存放着尚未计算出备份结点的结点对，它们的标记代表着其邻居是否有备份结点，即是否可以通过邻居结点到达目的结点，这里我们进行一个动态的更新：将标记值为1的结点对 (s,d) 弹出（第19－21行），利用路径生成算法PGA计算该结点对的备份路径（第22行）。$flag$ 用于标记该备份路径是否能到达目的结点，若备份路径可以到达目的结点，则 $flag$ 记为0，否则记为1。由于一个源-目的结点对可能有多条备份路径可以到达目的结点，因此需要选择一个最优备份下一跳存入字典，本文的选择规则为：当备份路径能够到达目的结点时（第23行），利用SBF算法判断备份路径中是否具有旁支结点，如果存在则优先将其对应的备份下一跳存入 $bn(s,d)$ 中；否则选择路径代价最小的备份结点添加至 $bn(s,d)$ 中（第24－29行）。当结点对 (s,d) 的备份下一跳计算完毕后，s 的邻居 $nei(s)$ 也可通过 s 结点到达 d，即若 R 中存在 $(nei(s),d)$，则将其标记值置为1，继续循环，直到 $R=\emptyset$ 或 R 中的sign值全为0（第30－32行）。如果仍有结点对尚无备份结点，则对这些结点对再次执行第二轮计算过程（第33－35行）。关于路径生成算法（PGA）和环路规避算法（LAA）是如何进行报文的转发与环路的规避，会在算法3及算法4中进行详细的阐述。

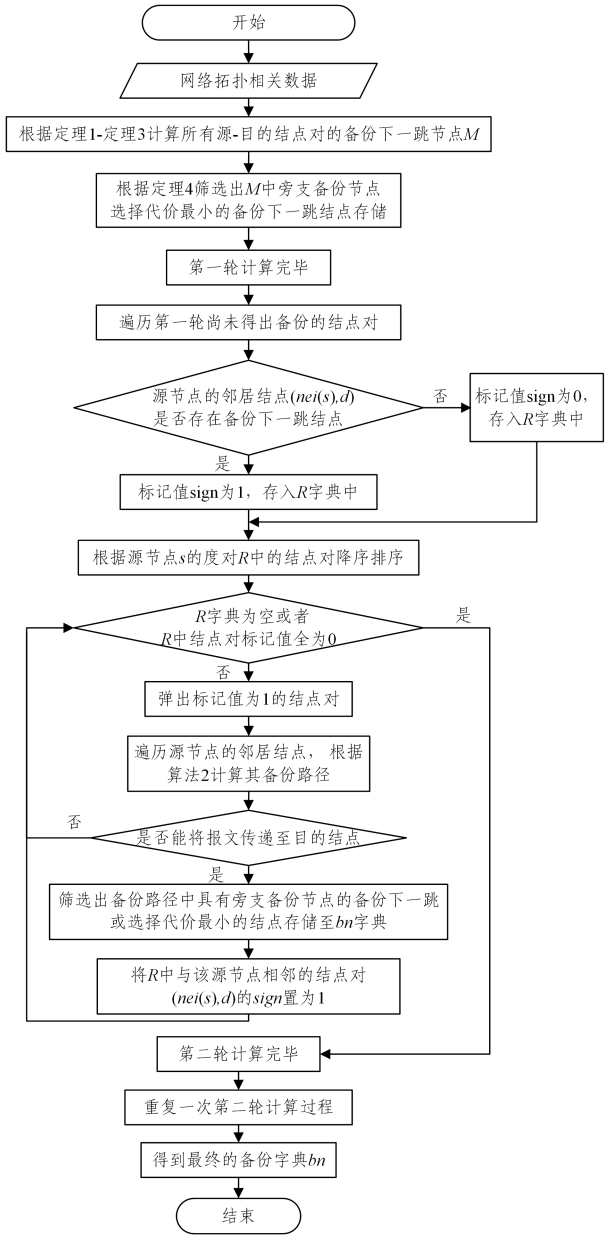

图2 RPAHFPR算法流程

Fig. 2 RPAHFPR algorithm flow

算法1 RPAHFPR

输入：G

输出：bn

```
1. FOR EACH (s,d) DO
2. M←Ø, bn←{}
3.     FOR EACH neighbor n of s DO
4.         IF level(d,s)=level(d,n) and father(d,s)!=father(d,n)
           THEN
5.             M←M∪{n}
6.         IF level(d,s)>level(d,n) THEN
7.             M←M∪{n}
8.         IF level(d,s)<level(d,n) and father(d,s) not in ance(d,n)
           THEN
9.             M←M∪{n}
10.        M←SBF(M,rspt(d),s,∀n∈M)
11.        path,flag←PGA(G,s,d,f)+LAA(node,path)
12.        bn(s,d)←arg min cost(path)
                      ∀n∈M
```

```
13.   END FOR
14. END FOR
15. FOR (s,d) WHICH bn(s,d)=Ø DO
16.   R(s,d)←(sign(s,d),degree(s))
17.     R.sort(degree)
18. END FOR
19. WHILE 1 exist in R.sign or R=Ø DO
20.     s,d←arg(sign=1)
21.     R.pop(s,d)
22.     path,flag←PGA(G,s,d,f)
23.     IF flag=0:
24.       side_node←SBF(path,rspt(d),s)
25.       IF side_node exists:
26.         bn(s,d)←nei(s)
27.         BREAK
28.       ELSE:
29.         bn(s,d)←arg min_{∀n∈nei(s)} cost(path)
30.       R(nei(s),d)←(1,degree(n))
31.   END IF
32. END WHILE
33. FOR(s,d) WHICH bn(s,d)=Ø DO
34.   EXECUTE lines 15-32 AGAIN
35. END FOR
```

5.2 旁支优先算法(SBF)

由定理4可知,对于一个结点对而言,旁支备份结点的备份路径能够完全避免再次遇到故障结点,具有很稳定的路由保护能力。因此本文提出SBF算法(见算法2)来筛选出可选备份结点中的旁支备份结点,使得在选择备份结点时能够优先选择旁支备份结点。算法2的输入为某结点对(s,d)的备份下一跳列表M、以d为根的反向最短路径树$rspt(d)$以及源结点s。输出为一个计算得出的备份下一跳M'。具体过程为:遍历M中每一个备份结点m,用*subroot*函数来计算每个备份结点对应的子树根结点(第1-3行)。如果m与源结点s的子树根结点不同,那么m就为旁支备份结点,将m加入M'中(第3-6行)。循环结束后,如果M'不为空,即存在旁支备份结点,则返回M',否则返回M(第7行)。

算法2 Side Branch First Algorithm(SBF)

```
输入:M,rspt(d),s
输出:M'
1. M'=Ø
2. FOR m in M:
3.   IF subroot(spt(d),m)≠subroot(rspt(d),s) THEN
4.     M'←M'∪m
5.   END IF
6. END FOR
7. RETURN M' or M
```

5.3 路径生成算法(PGA)

本节将详细介绍网络中报文的路径生成算法,理想的路径生成算法所计算出的转发路径不会产生环路且会优先选择代价最小的路径。算法3给出了PGA的执行过程。对于网络G中的任意一个源-目的结点对(s,d),设置一个列表*path*存放报文经过的结点,初始为空。*node*用来表示当前报文所在的结点,初始为源结点s,*flag*用于标记是否能够到达目的结点,*flag*初始为0,当无法到达目的结点时*flag*置为1(第1行)。在报文转发过程中,当前结点*node*永远指向报文所在的结点,当*node*到达目的结点时,*path*中会得到最终生成的备份路径(第8-10行)。循环过程中可能会遇到以下几种情况:

1)当前结点收到其最优下一跳结点发来的报文时,先将当前结点*node*加入*path*中(第2-3行),再将报文转发给其备份下一跳结点$bn(node,d)$,直到当前结点*node*的上一跳结点不是其最优下一跳结点(第4-6行)。

2)如果当前结点的最优下一跳*node.besthop*是故障结点,则将当前结点*node*加入*path*中;如果当前结点的备份下一跳存在且备份下一跳不会引发路由环路,则将报文转发给该备份下一跳,以该备份下一跳为源点递归调用*PGA*算法,否则标记*flag*为1,跳出循环(第11-18行)。

算法3 Path Generation Algorithm(PGA)

```
输入:G,s,d,f
输出:path
1. node←s,path←[],flag←0
2. WHILE node.lasthop=node.besthop DO:
3.     path.add(node)
4.     bn(node,d).lasthop←node
5.     node←bn(node,d)
6. END WHILE
7. WHILE TRUE:
8.     IF node.besthop=d THEN
9.       path.add(d)
10.        BREAK
11.    IF node.besthop=f THEN
12.      path.add(node)
13.      IF bn(node,d) exist and LAA(bn(node,d),path) THEN
14.        bn(node,d).lasthop←node
15.        PGA(bn(node,d),d,f,bestpath)
16.      ELSE
17.        flag←1
18.        BREAK
19.  ELSE
20.    path.add(node)
21.    node←node.besthop
22. END WHILE
23. IF flag=1 THEN
24.   FOR p in path DO
25.     backup←bn(p,d)
26.     IF subroot(rspt(d),backup)≠subroot(rspt(d),s) THEN
27.       path'←path[0,index(p)]
28.       path,flag←PGA(G,backup,d,f)
29.       BREAK
30.     END IF
31.   END FOR
32. END IF
33. RETURN path,flag
```

3)如果当前结点的最优下一跳不是故障结点,则将当前结点加入*path*中,将报文发送给其最优下一跳结点(第19-21行)。

4)跳出循环后,若*flag*值为1,则对*path*中的每一个结点进行检索,寻找每一个结点的备份结点是否是结点s的

旁支结点(第 23－25 行)。如果存在旁支结点，则计算旁支结点的位置，重新计算旁支结点所在位置之后的路径，利用 PGA 算法以旁支结点为源点 s 得到新的备份路径(第 26－33 行)。

5.4 环路规避算法(LAA)

为了解决路径生成算法中有可能出现的路由环路问题，本文提出了环路规避算法(LAA)。首先给出 LAA 算法的具体执行过程，再通过定理 5 和定理 6 从理论上证明其正确性。

算法 4 Loop Avoidance Algorithm(LAA)

输入：node，path

输出：TRUE or FALSE

```
1. IF node in path THEN
2.    IF node.lasthop≠node.besthop or path.count(node)≥2 THEN
3.       RETURN FALSE
4.    END IF
5. ELSE
6. RETURN TRUE
```

环路规避算法(见算法 4)的输入元素为 *node* 和 *path*，算法判断节点 *node* 是否会产生环路。在算法 3 中就是当前结点的备份下一跳结点，*path* 存放当前报文已经经过的结点，最终返回的是布尔类型值。具体过程为：当 *node* 结点已经存在于路径中时，如果结点 *node* 的上一跳 *node*.*lasthop* 不是其最优下一跳或者 *node* 结点在路径中的数量不小于 2 时，判断为出现环路(第 1－4 行)；如果 *node* 结点不存在于路径中，则判断为无环路(第 5－6 行)。

下面通过定理 5 和定理 6 来解释算法 4 第 2 行的含义并证明其正确性。

定理 5 已知 *path* 用于存储报文所经过的结点，*node* 为当前判断结点且 $node \in path$，如果 $node.lasthop \neq node.besthop$，则 *node* 会产生环路。

证明：利用反证法来证明其正确性。假设 $node \in path$ 且 $node.lasthop \neq node.besthop$ 不会产生环路。为便于读者理解，我们简化了图 1 的 $rspt(d)$ 得到如图 3 所示的 $rspt(d)'$。在图 3 中已知 $bn(g,d)=i$，当结点 a 发生故障时，计算(h,d)的备份路径，$path=(h,g,i,s,i)$，当 $bn(i,d)=g$ 时，则(i,d)的备份下一跳为 g，则有 $path=(h,g,i,s,i,g,h,g,i,s,i,\cdots)$，即结点 g 收到非最优下一跳结点 i 发来的报文出现了路由环路，与不会产生环路的结论矛盾，假设不成立，证毕。

定理 6 已知 *path* 用于存储报文所经过的结点，*node* 为当前判断结点且 $node \in path$，如果 $path.count(node) \geqslant 2$，则 *node* 会产生环路。

证明：利用反证法来证明其正确性。假设 $node \in path$ 且 $path.count(node) \geqslant 2$ 不会产生环路。在图 3 所示的$rspt(d)'$中计算(h,d)的备份路径，$bn(g,d)=i$，$bn(i,d)=g$。当结点 a 发生故障时，得到的备份路径为 $path=(h,g,i,s,i,g,h,g,i,s,\cdots)$，在 g 第三次出现时出现了路由环路，与结论矛盾，假设不成立，证毕。但如果 $bn(i,d)=e$，则当 $path=(h,g,i,s,i)$时，结点 i 可以将报文转发至结点 e 从而得到 $path=(h,g,i,s,i,e,b,d)$且不会产生环路。故当结点 i 第二次出现时并不一定会产生环路，第三次出现则一定会产生环路。

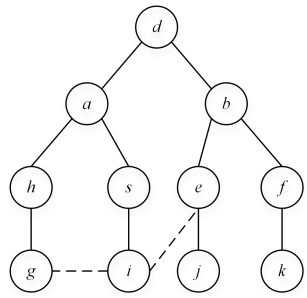

图 3 简化后的以结点 d 为根的反向最短路径树 $rspt(d)'$

Fig. 3 Simplified reverse shortest path tree $rspt(d)'$ rooted at node d

6 实验及结果分析

本文通过实验来模拟上述算法的性能，为更好地分析实验结果，我们将利用故障保护率和路径拉伸度对实验数据进行对比以充分论证本文算法的有效性。

6.1 网络拓扑

为了评估本文方法的性能，我们在大量拓扑上运行了该算法。实验中用到如下两种类型的网络拓扑。

1)真实网络拓扑[38]。我们从这类拓扑中选择了 11 个网络拓扑，具体参数如表 2 所列。

表 2 真实网络拓扑

Table 2 Real network topology

网络拓扑	结点数量	链路数量
Abilene	11	14
Cernet	14	16
TORONTO	25	55
USLD	28	45
Agis	16	21
Ans	17	24
Arpanet19719	18	22
Arpanet19723	24	27
Arpanet19728	29	32
AttMpls	25	56
Belnet2004	18	34

2)模拟拓扑。这类拓扑结构是用 Brite 模拟软件生成的拓扑结构，Brite 软件是由波士顿大学开发的一款生成拓扑的软件，其使用 Waxman 模型，路由器(结点)数量为 20～200，alpha 和 beta 的值分别设置为 0.15 和 0.2，网络的度设置在 2～12 之间，链路带宽是链路代价的倒数值。按照控制变量的原则，我们从这类拓扑中选择了 11 个网络拓扑，其中包含 5 个度都为 4、结点数量可变的拓扑和 6 个结点数量都为 200、度可变的拓扑，具体参数如表 3 所列。

表 3 Brite 网络拓扑

Table 3 Brite network topology

网络拓扑	结点数量	链路数量
Brite(20,4)	20	80
Brite(40,4)	40	160
Brite(60,4)	60	240
Brite(80,4)	80	320
Brite(100,4)	100	400
Brite(200,2)	200	400
Brite(200,4)	200	800
Brite(200,6)	200	1200
Brite(200,8)	200	1600
Brite(200,10)	200	2000
Brite(200,12)	200	2400

6.2 实验环境

本文基于 python 语言实现了 RPAHFPR 算法、NPC 算法以及 U-TURN 算法，并在上述拓扑网络中运行算法进行对比，运行实验的具体环境如表 4 所列。

表 4 实验运行环境

Table 4 Experimental operating environment

类别	配置
电脑类型	笔记本
显卡	NVIDIA GeForce GTX 960M
CPU	Intel Core i5-6300HQ
CPU 主频/GHz	2.30
内存大小/GB	12
操作系统	Windows 10
编程语言	Python 3.5.6

6.3 故障保护率

故障保护率是评估路由保护算法的一个重要指标，我们通过故障保护率来评估算法应对突发故障的能力，其定义如下。

定义 3 对于一个有 v 个结点的网络，其故障保护率指受保护的源-目的结点对数量 pn 与可保护结点对总数 $v*(v-1)-del$ 的比值，记为 FPR(Failure Protection Ratio)，用式(1)表示：

$$FPR=\frac{pn}{v*(v-1)-del}*100\% \tag{1}$$

其中，$v*(v-1)$ 表示结点对总数。由于路由器是逐跳转发的，因此对于一个源-目的结点对(s,d)，故障结点是 s 的最优下一跳，但如果最优下一跳就是目的结点 d，那么发生故障的结点就是 d。理论上，任何算法都无法规避该故障结点，因此在计算保护率时应当减去这样的结点对数量 del。由定义 1 可知，故障保护率指网络中被保护的源-目的结点对的数量占所有可保护结点对数量的比例。故障保护率为 100%时，网络中的所有源-目的结点对都被保护起来了，因此我们希望算法的故障保护率应当尽可能大，以提高算法对网络的保护能力。图 4 和图 5 分别给出了 3 种算法在不同网络拓扑中的故障保护率。

根据图 4 我们可以得出，在真实网络拓扑中，RPAHFPR 算法的故障保护率都接近于 100%，即可以应对绝大多数的网络单故障，在同样的拓扑网络中远高于 NPC 算法和 U-TURN 算法的故障保护率。同时 RPAHFPR 算法的故障保护率不会随着拓扑结构的变化而产生较大的变化，相比 NPC 和 U-TURN 算法具有很稳定的故障保护能力。

从图 5 中可以看出，在度都为 4 和拓扑大小为 200 的 Brite 模拟拓扑中，RPAHFPR 算法的故障保护率均接近 100%。而 NPC 算法的保护性能不稳定，在 200-2 的拓扑中故障保护率约为 91%，在其他拓扑中的故障保护率也均低于 RPAHFPR 算法。U-TURN 算法在 20-4 拓扑中的故障保护率约为 83%，在其他网络拓扑中的保护率也均低于 RPAHFPR。同时可以看出，RPAHFPR 算法的保护性能不会受限于网络拓扑的结构，具有稳定的保护能力，而 NPC 和 U-TURN 算法的保护性能则会受到网络拓扑结构的影响。因此，我们可以得出 RPAHFPR 算法在网络中应对故障的能力远高于 NPC 算法以及 U-TURN 算法。

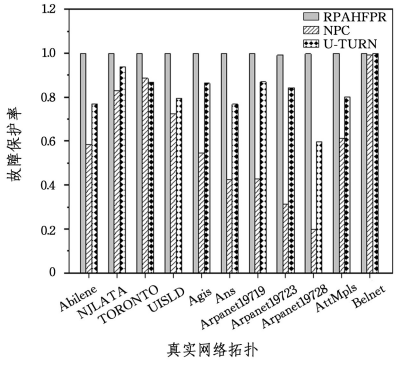

图 4 3 种算法在真实网络拓扑中的故障保护率

Fig. 4 Failure protection rates of three algorithms in real network topology

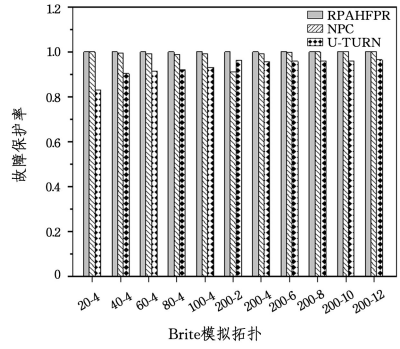

图 5 3 种算法在 Brite 模拟拓扑中的故障保护率

Fig. 5 Failure protection rate of three algorithms in Brite simulation topology

6.4 路径拉伸度

路径拉伸度是评估算法性能的另一个重要指标。在本文中，我们将路径拉伸度定义如下。

定义 4 假设一个网络有 v 个结点，删除网络中任意一个结点 v_i，然后利用 RPAHFPR 算法计算网络中所有源-目的结点对的路径代价之和，记为 $S(v_i)$；利用 OSPF 算法计算网络中所有源-目的结点对的路径代价之和，记为 $K(v_i)$。分别累加删除每一个结点后所对应的 $S(v_i)$和 $K(v_i)$，$S(v_i)$与$K(v_i)$累加和的比值即为本文算法的路径拉伸度，记为 PS(Path Stretch)，如式(2)所示：

$$PS=\frac{\sum_{i=1}^{v}S(v_i)}{\sum_{i=1}^{v}K(v_i)} \tag{2}$$

路径拉伸度描述了执行某个算法所花费的路径代价与默认的 OSPF 算法所花费的路径代价的比值，在路由保护中用于衡量算法的路径代价开销。显然，在实际应用中，当一个结点发生故障时，所有最优路径中经过该故障结点的源-目的结点对都会受到影响。对于一个网络拓扑而言，每一个结点都有可能出现故障，发生故障的结点是随机且无法预知的。鉴于此，本定义考虑了网络中的每一个结点发生故障时，对整个网络造成的影响，符合实际应用需要，计算结果具有代表性。

此外，由于 RPAHFPR 算法的故障保护率远高于 NPC 算法和 U-TURN 算法，即对于同一个拓扑，RPAHFPR 算法

所保护的结点对数量远多于 NPC 算法和 U-TURN 算法。在计算一个算法的路径拉伸度时，如果按照其自身算法得到的所有受保护的结点对进行代价的计算显然是不合理的，对于保护率高的算法而言，这种方式会多算一部分结点对的拉伸度。因此本文选择两种算法所保护的结点对的交集进行代价计算，通过两两比较得到 RPAHFPR 算法与 NPC 算法和 U-TURN 算法的路径拉伸度，计算结果如表 5 和表 6 所列。

表 5 RPAHFPR 算法和 NPC 算法在不同拓扑的路径拉伸度

Table 5 Path stretch degree of RPAHFPR and NPC algorithms in different topologies

拓扑类型	拓扑名称		RPAHFPR	NPC
真实拓扑结构	Abilene		1.0107	1.0080
	NJLATA		1.0085	1.0097
	TORONTO		1.0229	1.0154
	USLD		1.0114	1.0051
	Agis		1.0169	1.0014
	Ans		1.0213	1.0029
	Arpanet19719		1.0061	1.0013
	Arpanet19723		1.0109	1.0004
	Arpanet19728		1.0145	1.0001
	AttMpls		1.0104	1.0038
	Belnet2004		1.0000	1.0000
Brite 模拟拓扑结构	度为 4	20-4	1.0053	1.0188
		40-4	1.0137	1.0151
		60-4	1.0122	1.0122
		80-4	1.0122	1.0131
		100-4	1.0139	1.0116
	结点大小为 200	200-2	1.0074	1.0040
		200-4	1.0089	1.0069
		200-6	1.0064	1.0069
		200-8	1.0055	1.0067
		200-10	1.0041	1.0058
		200-12	1.0035	1.0056

表 6 RPAHFPR 算法和 U-TURN 算法在不同拓扑的路径拉伸度

Table 6 Path stretch degree of RPAHFPR and U-TURN algorithms in different topologies

拓扑类型	拓扑名称		RPAHFPR	NPC
真实拓扑结构	Abilene		1.0256	1.0211
	NJLATA		1.0107	1.0025
	TORONTO		1.0256	1.0043
	USLD		1.0246	1.0089
	Agis		1.0311	1.0210
	Ans		1.0327	1.0186
	Arpanet19719		1.0395	1.0329
	Arpanet19723		1.0373	1.0325
	Arpanet19728		1.0291	1.0183
	AttMpls		1.0138	1.0023
	Belnet2004		1.0000	1.0005
Brite 模拟拓扑结构	度为 4	20-4	1.0039	1.0011
		40-4	1.0115	1.0013
		60-4	1.0106	1.0021
		80-4	1.0101	1.0026
		100-4	1.0116	1.0024
	结点大小为 200	200-2	1.0087	1.0034
		200-4	1.0078	1.0017
		200-6	1.0055	1.0009
		200-8	1.0049	1.0006
		200-10	1.0036	1.0003
		200-12	1.0032	1.0002

从表 5 可以看出，在不同网络拓扑中 RPAHFPR 算法的路径拉伸度接近于 1，即相较于默认的 OSPF 修复故障的代价，RPAHFPR 算法不会带来庞大的路径代价开销。此外，RPAHFPR 算法和 NPC 算法的路径拉伸度相差很小，在部分拓扑上 RPAHFPR 算法的拉伸度甚至小于 NPC 算法，这表明 RPAHFPR 算法与 NPC 算法的路径代价消耗是一个数量级别，且都接近于默认的 OSPF 代价，但 RPAHFPR 算法的故障保护能力却远高于 NPC 算法。

表 6 中的结果表明，在不同网络拓扑中 RPAHFPR 算法的路径拉伸度均接近于 1，具有较小的路径开销，同时，RPAHFPR 算法与 U-TURN 算法所带来的路径开销十分接近，但 RPAHFPR 算法的故障保护率却远高于 U-TURN 算法。

结束语 针对现有的路由保护算法故障保护率不高的问题，本文提出了一种基于 SDN 的高故障保护率的路由保护方案 RPAHFPR，该方案利用 SDN 集中控制的优势，对网络中所有的结点对进行备份下一跳结点的计算。为得到无环稳定的备份结点，本文方案在计算过程中融合了环路规避算法、旁支优先算法以及路径生成算法。实验结果表明，相比 NPC 算法和 U-TURN 算法，RPAHFPR 算法具有更高的故障保护率且不会带来庞大的路径开销，能够应对网络中绝大多数的路由单故障。但是 RPAHFPR 算法只能应对 SDN 网络中的单结点或单链路故障，无法解决并发故障问题。下一步我们将重点研究 SDN 网络中并发故障的路由保护方案，提高网络应对并发结点故障的自愈性。

参 考 文 献

[1] ANTONAKOPOULOS S, KOPPOL P, et al. Full protection made easy: the dispath ip fast reroute schemep[J]. IEEE/ACM Transactions on Networking: A Joint Publication of the IEEE Communications Soceity, the IEEE Computer Society, and the ACM with Its Special Interest Group on Data Communication, 2015, 23(4): 1229-1242.

[2] ANBIAH A, SIVALINGAM K M. Efficient failure recovery techniques for segment-routed networks[J]. Computer Communications, 2022, 182: 1-12.

[3] TAPOLCAI J, RETVARI G, BABARCZI P, et al. Scalable and efficient multipath routing: complexity and algorithms[C]// 2015 IEEE 23rd International Conference on Network Protocols (ICNP). 2016: 376-385.

[4] XU M W, LI Q, PAN L T, et al. Network Failure and Self-healing Routing Model and Algorithm[J]. SCIENTIA SINICA Informationis, 2010, 40(7): 943-953.

[5] YUAN Y, XU M, QI L. Tunneling on demand : a lightweight approach for ip fast rerouting against multi-link failures[C]// 2016 IEEE/ACM 24th International Symposium on Quality of Service(IWQoS). Beijing: IEEE, 2016: 1-6.

[6] GOPALAN A, RAMASUBRAMANIAN S. IP fast rerouting and disjoint multipath routing with three edge-independent spanning trees[J]. IEEE/ACM Transactions on Networking, 2016, 24(3): 1336-1349.

[7] LABOVITZ C, AHUJA A, BOSE A, et al. Delayed internet routing convergence[J]. IEEE/ACM Transactions on Networking, 2001, 9(3): 293-306.

[8] FORTZ B, THORUP M. Optimizing ospf/is-is weights in a changing world[J]. IEEE Journal on Selected Areas in Communications, 2002, 20(4): 756-767.

[9] CLAD F, MERINDOL P, VISSICCHIO S, et al. Graceful router updates in link-state protocols [C]// 21st IEEE International Conference on Network Protocols. Goettingen: IEEE, 2013: 1-10.

[10] ASTUTO B N, CA M M, XUAN N N, et al. A survey of software-defined networking: past, present, and future of programmable Networks[J]. IEEE Communications Surveys & Tutorials, 2014, 16(3): 1617-1634.

[11] AMIN R, REISSLEIN M, SHAH N. Hybrid sdn networks: a survey of existing approaches[J]. IEEE Communications Surveys & Tutorials, 2018, 20(4): 3259-3306.

[12] YANG Y, XU M, LI Q. Fast rerouting against multi-link failures without topology constraint[J]. IEEE/ACM Transactions on Networking, 2018, 26(1): 384-397.

[13] LEE S, YU Y, NELAKUDITI S, et al. Proactive vs reactive approaches to failure resilient routing[C]// Joint Conference of the IEEE Computer & Communications Societies. Hong Kong: IEEE, 2004: 176-186.

[14] KWONG K W, GAO L, GUERIN R, et al. On the Feasibility and Efficacy of Protection Routing in IP Networks[J]. IEEE/ACM Transactions on Networking, 2011, 19(5): 1543-1556.

[15] ATLAS A, ZININ A. Basic specification for ip fast reroute: Loop-free alternates[R]. Heise zeitschriften verlag, 2008.

[16] BRYANT S. Ip fast reroute framework. LoopFree Alternatesn [S]. USA: The Internet Engineering Task Force, 2010.

[17] BRYANT S, PREVIDI S, SHAND M. A framework for ip and mpls fast reroute using not-via addresses[S]. USA: Internet Requests for Comments, RFC, 2013.

[18] RÉTVÁRI G, TAPOLCAI J, ENYEDI G, et al. Ip fast reroute: loop free alternates revisited[C]// 2011 Proceedings IEEE INFOCOM. Shanghai: IEEE, 2011: 2948-2956.

[19] CSIKOR L, NAGY M, RÉTVÁRI G. Network optimization techniques for improving fast ip-level resilience with loop-free alternates[J]. Infocommunications Journal, 2011, III(4): 2-10.

[20] RÉTVÁRI G, CSIKOR L, TAPOLCAI J, et al. Optimizing igp link costs for improving ip-level resilience[C]// 8th International Workshop on the Design of Reliable Communication Networks. Krakow: DRCN, 2011: 62-69.

[21] CSIKOR L, RÉTVÁRI G. On providing fast protection with remote loop-free alternates [J]. Telecommunication Systems, 2015, 60(4): 1-18.

[22] BRYANT S, FILSFILS C, PREVIDI S, et al. Remote loop-free alternate(lfa) fast reroute(fpr)[S]. USA: The Internet Engineering Task Force, 2015.

[23] GENG H J, ZHANG H, SHI X G, et al. Efficient computation of loop-free alternates[J]. Journal of Network and Computer Applications, 2020, 151(102501): 1-12.

[24] KAMISINSKI A. Evolution of ip fast-reroute strategies[C]// 2018 10th International Workshop on Resilient Networks Design and Modeling. Longyearbyen: RNDM, 2018: 1-6.

[25] TAVERNIER W, PAPADIMITRIOU D, COLLE D, et al. Self-configuring loop-free alternates with high link failure coverage [J]. Telecommunication Systems, 2014, 56(1): 85-101.

[26] DECRAENE B, FRANCOIS P, BASHANDY A, et al. Topology independent fast reroute using segment routin[S]. USA: The Internet Engineering Task Force, 2014.

[27] CIANFRANI A, LISTANTI M, POLVERINI M. Incremental Deployment of Segment Routing Into an ISP Network: a Traffic Engineering Perspective[J]. IEEE/ACM Transactions on Networking, 2017, 25(5): 3146-3160.

[28] ATLAS A et al. U-turn alternates for ip/ldp fast-reroute[S]. USA: The Internet Engineering Task Force, 2006.

[29] ENYEDI G, SZILAGYI P, RETVARI G, et al. Ip fast reroute: lightweight not-via without additional addresses[C]// IEEE INFOCOM 2009. Rio de Janeiro: IEEE, 2009: 2771-2775.

[30] ZHANG H L, XIAO G, YAN J Y, et al. Sdn-based ecmp algorithm for data center networks[C]// 2014 IEEE Computing, Communications and IT Applications Conference(ComComAp). Beijing: IEEE, 2014: 13-18.

[31] BRAUN W, MENTH M, et al. Loop-free alternates with loop detection for fast reroute in software-defined carrier and data center networks[J]. Journal of Network & Systems Management, 2016, 24(3): 470-490.

[32] ZHANG X, CHENG Z, LIN R P, et al. Local fast reroute with flow aggregation in software defined networks[J]. IEEE Communications Letters, 2017, 21(4): 785-788.

[33] QIU K, ZHAO J, WANG X, et al. Efficient recovery path computation for fast reroute in large-scale software defined networks[J]. IEEE Journal on Selected Areas in Communications, 2019, 37(8): 1755-1768.

[34] THORAT P, CHALLA R, RAZA S M, et al. Proactive failure recovery scheme for data traffic in software defined networks [C]// 2016 IEEE Netsoft Conference & Workshops. Seoul: IEEE, 2016: 219-225.

[35] ZHENG J, XU H, ZHU X, et al. Sentinel: failure recovery in centralized traffic engineering[J]. IEEE/ACM Transactions on Networking, 2019, 27(5): 1859-1872.

[36] GENG H J, ZHANG W, YIN X. Routing protection algorithm based on hybrid software defined network[J]. Computer Engineering, 2020, 46(6): 209-215.

[37] GENG H J, WANG W, YIN X. Single-node fault protection method based on hybrid software-defined network[J]. Computer Science, 2022, 49(2): 329-335.

[38] GP RÉDEI. Abilene(advanced networking for leading-edge research and education)[M]. Berlin: Springer Netherlands, 2008.

GENG Haijun, born in 1983, Ph.D, associate professor. His main research interests include software defined network, Internet routing algorithm and network architecture.

(责任编辑:何杨)