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.

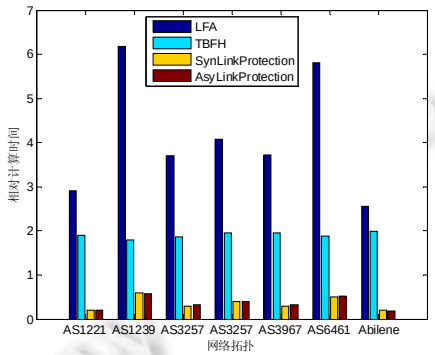

Fig.2 Computation overhead with link protection in real and measured topologies

图 2 真实和测量拓扑中链路保护计算开销

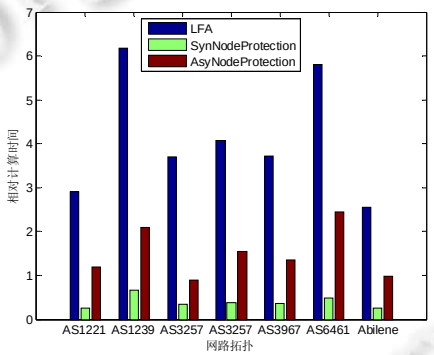

Fig.3 Computation overhead with node protection in real and measured topologies

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.

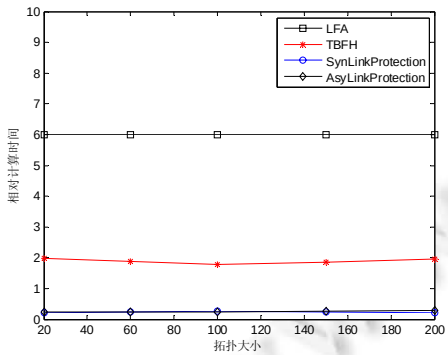

Fig.4 Computation overhead with link protection in generated topologies

图 4 Brite 生成拓扑中链路保护计算开销

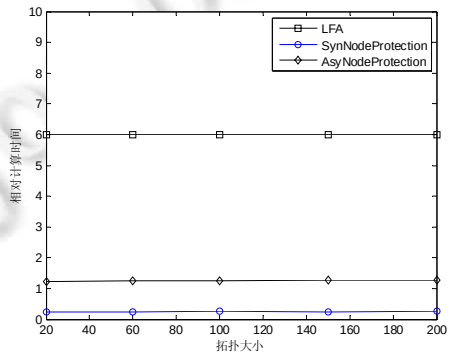

Fig.5 Computation overhead with node protection in generated topologies

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.

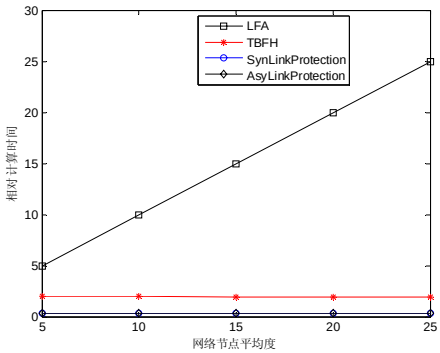

Fig.6 Computation overhead with link protection in generated topologies

图 6 Brite 生成拓扑中链路保护计算开销

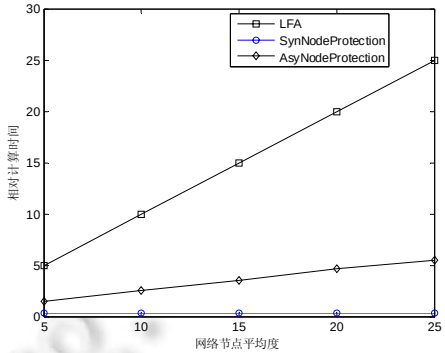

Fig.7 Computation overhead with node protection in generated topologies

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.

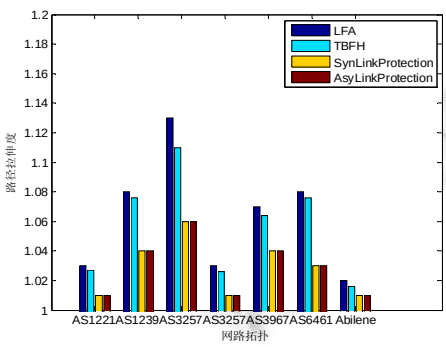

Fig.8 Path Stretch with link protection in real and measured topologies

图 8 真实和测量拓扑中链路保护路径拉伸度

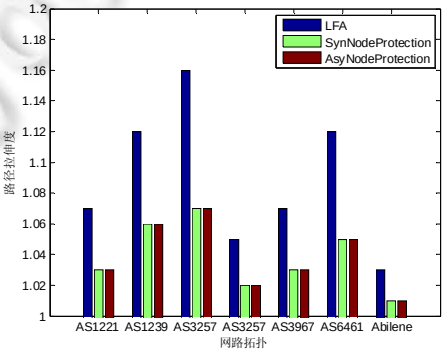

Fig.9 Path Stretch with node protection in real and measured topologies

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.

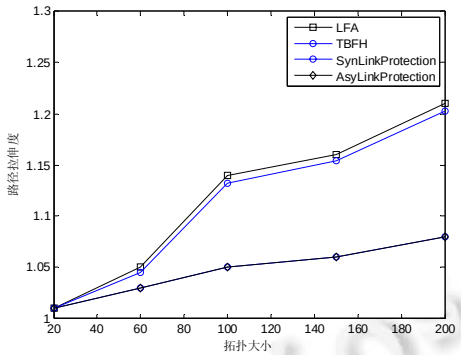

Fig.10 Path Stretch with link protection in generated topologies

图 10 Brite 生成拓扑中链路保护路径拉伸度

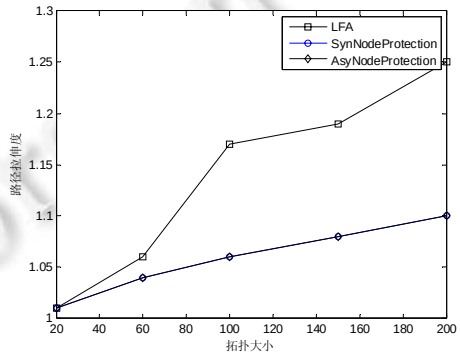

Fig.11 Path Stretch with node protection in generated topologies

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.

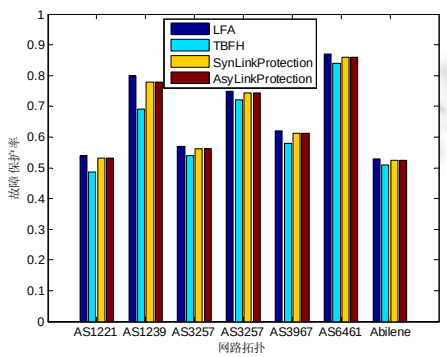

Fig.12 Protection rate with link protection in real and measured topologies

图 12 真实和测量拓扑中链路保护故障保护率

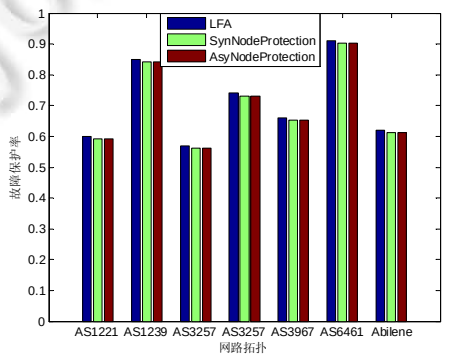

Fig.13 Protection rate with node protection in real and measured topologies

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.

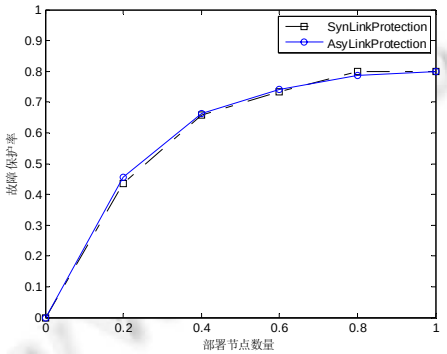

Fig.14 Deployment with link protection in Sprint topology

图 14 Sprint 拓扑中链路保护部署情况

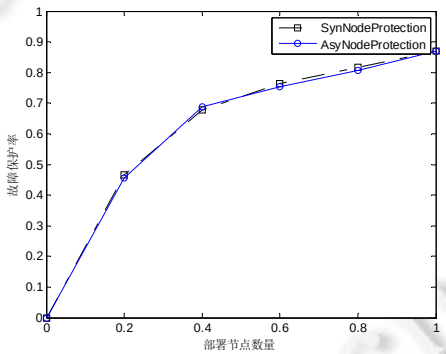

Fig.15 Deployment with node protection in Sprint topology

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.