基于逐跳方式的单链路故障保护算法

耿海军[1] 施新刚[2] 王之梁[2] 尹 霞[2] 尹少平[1]
(山西大学软件学院 太原 030013)[1] (清华大学计算机科学与技术系 北京 100048)[2]

摘 要 当网络中的某条链路出现故障时,互联网部署的域内路由协议需要重新收敛,在收敛过程中经过该链路的报文将会被丢弃。针对该问题,IETF(the Internet Engineering Task Force)提出了快速重路由保护框架,利用该框架可以有效地解决网络中单链路故障造成的报文丢失问题,然而该方案并不能完全保护网络中所有可能的单链路故障。基于该框架研究者提出了一种基于隧道的解决方案,该方案虽然可以提供100%的单链路故障保护,但是需要辅助机制的协助,开销较大,难以实际部署。因此,提出了一种基于逐跳方式的针对单链路故障的全保护方案,该方案可以解决网络中任意的单链路故障造成的报文丢失问题。

关键词 路由保护,链路故障,域内路由
中图法分类号 TP309.7 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.07.012

Single-link Failure Protection Algorithm Based on Hop-by-Hop Routing

GENG Hai-jun[1] SHI Xin-gang[2] WANG Zhi-liang[2] YIN Xia[2] YIN Shao-ping[1]
(School of Software Engineering, Shanxi University, Taiyuan 030013, China)[1]
(Department of Computer Science & Technology, Tsinghua University, Beijing 100048, China)[2]

Abstract When a link fails, the deployed intra-domain routing protocol need to re-convergen. During the process of re-convergence, packets which travel through the link will be dropped. Therefore, IETF proposed a framework, named IP-FRR (IP Fast Re-Route), which can effectively reduce packet loss when a single-link failure occurs. However, IPFRR cannot provide complete protection against all single-link failure scenarios. Based on the framework, lots of IP-tunnel-based schemes have been proposed to solve the problem. However, these methods introduce extra overhead and need the assistance of an auxiliary mechanism. In this work, we aimed for hop-by-hop routing algorithm that can provide complete protection against all such single-link failure scenarios.

Keywords Routing protection, Link failure, Intra-domain routing

互联网在50年的发展过程中遇到了很多问题,其中部分问题严重影响了网络的运行效率和性能,大大降低了用户对网络的体验程度。对于网络中的域内路由而言,其中路由可用性是一个严重影响其性能的重要问题。因此,域内路由可用性研究是互联网路由急需解决的重大科学问题之一。现在部署的域内路由协议收敛时间较长[1],通常需要几秒甚至十几分钟的时间,其间产生的路由黑洞和路由环路会导致网络中断,从而降低网络可用性。目前,已经有很多实时应用出现在互联网上[2-4],它们对网络性能的要求更加严格[5-10],通常需要在50ms内完成路由的收敛过程。如果路由收敛时间超出50ms,这些应用的性能将会受到严重的影响,大大降低用户的体验效果。这些新型应用对网络的时延和可用性提出了更加严格的要求,因此当前路由协议遭遇故障后的自动恢复能力[11]将会受到极大挑战。相比域内路由协议采用的被动收敛机制,路由保护可以很好地解决由于网络故障造成的报文丢失问题,大大降低了由于网络中断造成的财产损失和用户体验度。因为路由保护方案可以灵活应对网络中的故障,并且在互联网中实现相对容易,因此该机制受到了学术界和工业界的支持和关注。

在互联网领域,目前工业界采用的转发报文的方式主要有两种类型,非逐跳转发和逐跳转发。非逐跳转发方式需要一些辅助方案的配合才能正常工作,引入了较大的开销,因此部署较为困难,不适合在实际中使用。逐跳发方式转比较简单,并不需要额外方案的协助,因此该转发方式得到了因特网服务提供商的大力支持,从而得到了广泛部署。Not-Via[12]和源路由[13]是两种比较典型的采用非逐跳方式转发报文的路由协议。为了保护网络中的链路,Not-Via在报文的头部增加了特殊的地址,从而保证报文在转发过程中可以有效地

到稿日期:2016-05-09 返修日期:2016-07-31 本文受国家高技术研究发展计划(863计划)基金资助项目(2015AA015603,2015AA016105),国家自然科学基金资助项目(61402253)资助。

耿海军(1983-),男,博士,讲师,主要研究方向为网络体系结构和路由算法等,E-mail:ghj123025449@163.com;**施新刚**(1980-),男,高级工程师,主要研究方向为路由算法、SDN网络等,E-mail:shixg@cernet.edu.cn;**王之梁**(1978-),男,副研究员,主要研究方向为网络体系结构、网路协议测试等,E-mail:wzl@cernet.edu.cn。

绕过故障链路，保证报文转发的正确性。源路由在报文的头部显式地标明转发报文转发的整条路径，但是该方案需要大量的存储空间。上述两种方案都不适合在互联网中实际部署。ECMP[14]、LFA[15]、偏转路由[16]和 U-turn[17]都是采用逐跳方式转发报文的路由协议。等价多路径路由（Equal Cost Multiple Paths，ECMP）是目前域内路由协议中部署的一种多路径路由协议，但是相关研究[18]表明，在 ECMP 中实现最优的路由算法是一个 NP-Hard 难题。基于该问题，研究者提出利用 LFA 来提高网络中路由可用性，LFA 利用一些无环路规则提前计算出备份路径，当网络中真正出现故障时，可以利用这些备份路径转发受影响的报文。然而研究表明，LFA 的故障保护率较低，仅仅在 50%[19-20]左右。为了进一步提高 LFA 的可用性，文献[16-17]分别提出了新的解决方案。偏转路由提出了更加复杂度的无环路规则，利用这些规则虽然可以计算出更多的备份路径，但是算法复杂度较高。U-turn 将保护下一跳延伸到默认下一跳的下一跳，可以保护更多的链路，从而增加了路由可用性。

文献[21]表明，网路中的故障频繁发生，其中 70%的故障为单链路故障，因此本文主要关注网络中单链路故障情形。然而已有的非逐跳转发方案都无法提供 100%的单链路故障保护率，因此本文提出了一种基于逐跳方式的针对单链路故障的全保护方案。本文提出的方案可以 100%保护网络中的所有单链路故障，然而该方案并不能 100%保护网络中所有的单节点故障和并发故障，因此，下一步将重点研究单节点和并发故障对应的全保护方案。

1　单链路故障保护算法

1.1　网络模型和基本理论

网络可以抽象为图 $G=(V,E)$，其中 V 表示节点（路由器）的集合，E 表示边（链路）的集合。对于图中的任意一条边 $e=(x,y)\in E$，用 $w(e)$ 表示该边对应的权值。用 $Primary(s,d)$ 和 $Backup(s,d)$ 分别表示源节点 s 到目的节点 d 的最优下一跳和备份下一跳；$SP(s,d)$ 表示从节点 s 到节点 d 最短路径包含的链路；$Cost(s,d)$ 表示从节点 s 到节点 d 最短路径的代价。

定义 1　对于任意目的节点 d，如果 $E_d\subset E$，$|E_d|=|V|-1$，并且任意节点 $v\in V$ 到目的节点 d 都拥有最小代价，则称 $T_d(V,E_d)$ 表示以 d 为根的最短路由树。在路由树 T_d 中，对于任意节点 $v\in V$，用 $child(v)$ 表示节点 v 的孩子节点，$parent(v)$ 表示节点 v 的父亲节点，$subtree(d,v)$ 表示在 T_d 中以节点 v 为根的子树。

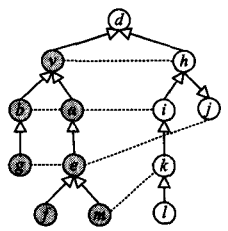

图 1　以节点 d 为根的最短路由树

下面通过一个例子来解释上述定义，图 1 中所有的实线和虚线构成了一个网络图，实线表示以节点 d 为根的最短路由树。源节点 f 到目的节点 d 的最短路径记为 $SP(f,d)=(f,e,a,v,d)$，源节点 f 到目的节点 d 的最优下一跳记为 $Primary(f,d)=e$。节点 v 有两个孩子节点 $child(a)=\{a,b\}$。实心节点表示以节点 v 为根的子树，用 $subtree(d,v)$ 表示。

定义 2　对于任意一个事件(s,d,f)，其中 s 表示源节点，d 表示目的节点，f 表示路径 $SP(s,d)$ 上的单链路故障。当该事件发生时，如果 s 到 d 的路径依然保持连通，则称该网络具有健壮的网络拓扑结构。

定义 3　在以 d 为根的最短路由树中，对于事件(v,d,f)，其中 f 表示路径 $SP(v,d)$ 上的单链路故障，如果存在一条链路(x,y)，使得 $x\in subtree(d,v)$ 和 $y\in V-subtree(d,v)$ 同时成立，则称链路(x,y)为子树 $subtree(d,v)$ 的桥，用 $Candidate(v)=\{(x,y)\}$ 表示。

定理 1　对于一个健壮的网络拓扑结构，对于事件(v,d,f)，其中 f 表示路径 $SP(v,d)$ 上的单链路故障，则子树 $subtree(d,v)$ 至少有一个桥。

证明：使用反证法。假设不存在任何链路(x,y)使得 $x\in subtree(d,v)$ 和 $y\in V-subtree(d,v)$ 同时成立，那么对于任意节点 $x\in subtree(d,v)$，与节点 x 相连的链路的另一端 y 仅仅和集合 $subtree(d,v)$ 中的节点相连，y 到 d 的最短路径必定经过节点 v。则当节点 v 到目的节点 d 的默认下一跳出现故障时，v 和 d 之间将无法连通，这与前提假设相矛盾。即原假设不成立，定理得证。

图 1 形象地表示了定理 1，利用 T_d 可得 $subtree(d,v)$（图 1 中实心节点）。定理 1 说明了如何寻找子树 v 对应的桥，遍历集合 $x\in subtree(d,v)$ 中的所有邻居节点，找到其所有在 $y\in V-subtree(d,v)$ 中的邻居节点，两个集合中的节点连接的边即为子树 $subtree(d,v)$ 对应的桥。在图 1 中，节点 v 的子树 $subtree(d,v)$ 所对应的桥为 $Candidate(v)=\{(v,h),(a,i),(e,j),(m,k)\}$。

由上述例子可知，节点的子树对应的桥并不是唯一的，不同桥对应的保护路径的代价是不同的。下面将重点解决 3 个问题：1）如何找出子树对应的所有桥；2）如何选择最佳的桥，从而使得保护路径具有最小的代价；3）如何为节点计算保护下一跳。

由于对于任意目的节点的计算方法都是类似的，因此，不失一般性，算法仅考虑目的地址为 d 的计算方法。下面介绍如何解决上述 3 个问题。根据深度优先算法遍历以 d 为根的最短路由树中的所有节点，当访问某个节点 v 时，假设该节点和其最优下一跳之间的链路出现故障，将子树 $subtree(d,v)$ 中的所有节点标记为实心节点，遍历子树 $subtree(d,v)$ 中的所有节点，对于该子树中的节点 x，检查其每一个邻居节点 y，如果该节点不是实心节点，则链路(x,y)为子树 $subtree(d,v)$ 的桥。由式(1)计算保护路径的代价。

$$Cost(v,x)+Cost(x,y)+Cost(y,d) \tag{1}$$

式(1)中涉及到的变量都可以很容易地从链路状态路由协议中得到。其中 $Cost(x,y)$ 可以从链路状态数据库中得到，其余变量可以通过 T_d 得到，因此可以计算相应的桥对应的保护路径的代价。对于找到的所有桥，选择其中具有最短

保护路径的一个作为最终的桥。最后为相应的节点计算保护下一跳。根据选定的桥，计算节点 v 的重路由路径，假设该路径为 $(v,m_1,m_2,\cdots,m_k,x,y)$，则相应节点的保护下一跳为：$Backup(v,d)=m_1$，$Backup(m_1,d)=m_2,\cdots,Backup(x,d)=y$。

1.2 路由保护算法

算法 1 描述了如何为节点的子树计算对应的桥、如何选择最终桥以及如何为节点计算保护下一跳。对于目的节点 d，根据深度优先算法遍历以 d 为根的最短路由树中的所有节点，当访问某个节点 v 时，假设该节点和最优下一跳之间的链路出现故障。首先，将子树 $subtree(d,v)$ 中的所有节点标记为实心节点，遍历子树 $subtree(d,v)$ 中的所有节点，找到具有最短保护路径的一个桥作为最终的桥，记为 (m,n)。最后为相应的节点计算保护下一跳。

算法 1 Link-protection(d)

```
//每个节点计算到目的地址 d 的备份下一跳
Input:
G(V,E),目的地址 d
Output:
Backup[][d]
1. for v∈V do
2. Backup[v][d]← ∅;
3. v.color ← white;
4. 计算 Td(V,Ed),并且存储节点到 d 的最优下一跳;
   //按照深度优先顺序访问 Td(V,Ed)中的节点;对于每个节点 v,假设该节点到 d 的最优下一跳链路出现故障
5. for v∈V,v≠d do
6. f ← Primary(v,d);
7. 将 subtree(d,f)中的所有节点标记为实心节点;
8. if Backup[][d]≠∅ then
9. continue;
10. mincost ← ∞;
11. for x∈subtree(d,f) do
12. for 每个与节点 v 直接相连的节点 y do
13. if y 是非实心节点 or
    Cost(v,x)+Cost(x,y)+Cost(y,d)−Cost(v,d)≥ mincost then
14. continue;
15. mincost ←Cost(v,x)+Cost(x,y)+Cost(y,d)−Cost(v,d)
16. AssignBackup(x,y,v,d);
17. 将所有节点标记为非实心节点;
18. return Backup[][d]
```

算法 2 AssignBackup(x,y,v,d)

```
//为路径 SP(x,v)上的所有节点计算备份下一跳
1. u ← x;
2. v ← y;
3. while u≠v do
4. if Backup[u][d]=∅ then
5. Backup[u][d]← v;
6. v ← u;
7. u ← Primary(u);
```

图 2 描述了为路径 (a,c,e,f) 中的节点计算备份下一跳的过程，图中圆圈中的数值表示该节点到目的节点 d 的最小代价。

(1)假设当链路 (a,d) 断开时，找出所有以 a 为根的子树的桥 $\{(a,k),(c,g),(e,g),(f,i),(f,j)\}$。当桥为 (c,g) 时，对应的保护路径的代价最小，此时节点 a 对应的重路由路径为 (a,c,g)，因此 $Backup[c][d]=g$，$Backup[a][d]=c$。

(2)假设当链路 (c,a) 断开时，因为节点 c 已经找到备份下一跳，因此不执行任何操作。

(3)假设当链路 (e,c) 断开时，找出所有以 e 为根的子树的桥 $\{(e,b),(e,g),(f,i),(f,j)\}$，当桥为 (e,b) 时，保护路径的代价最小，此时节点 e 的重路由路径为 (e,b)，因此 $Backup[e][d]=b$。

(4)假设当链路 (f,e) 断开时，找出所有以 f 为根的子树的桥 $\{(f,i),(f,j)\}$，当桥为 (f,j) 时，保护路径的代价最小，此时节点 f 的重路由路径为 (f,j)，因此 $Backup[f][d]=j$。

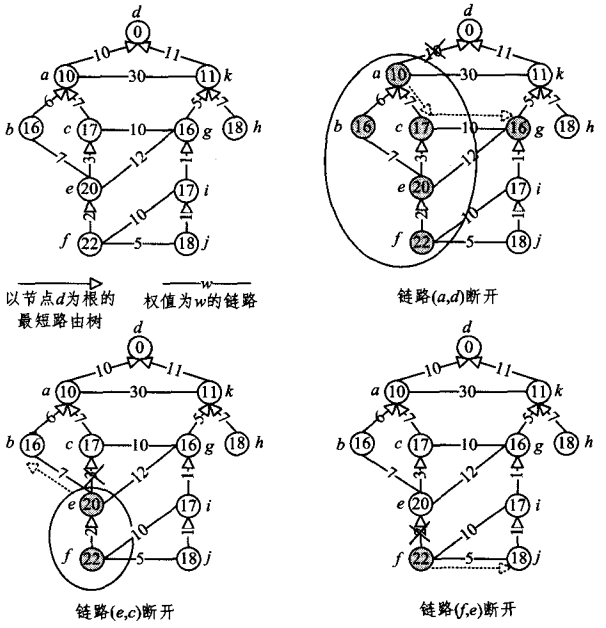

图 2 计算备份下一跳实例

定理 2 当网络中的所有节点都部署上述的路由保护算法时，网络中任意单链路故障即可被保护。

证明：对于网络中任意的源节点 s 和目的节点 d，假设链路 $L=(x,y)\in SP(s,d)$ 出现故障。根据定理 1 可知，子树 $subtree(d,x)$ 至少存在一个桥，因此算法可以保护链路 L。由于节点 s 是节点 x 的子孙节点，因此当链路 $L=(x,y)$ 断开时，从源节点 s 发送到目的节点 s 的报文可以正常到达。

定理 3 算法 1 的时间复杂度为 $|V|O(|V|+|E|)$。

证明：在为某个节点计算保护下一跳的算法中(第 11—15 行)，网络中的每个节点最多被访问一次，而每条链路最多被访问两次，因此为每个节点计算保护下一跳的时间复杂度为 $O(|V|+|E|)$。如果利用算法 1 为网络中所有节点计算保护下一跳，则该算法的时间复杂度为 $|V|O(|V|+|E|)$。

2 分布式方案

以上算法采用集中式计算，可以应用在各种网络协议中，例如距离矢量路由协议和链路状态路由协议。如果采用分布式算法，上述算法只能应用在链路状态路由协议中，例如 OS-

PF，IS-IS 等。下面将分别介绍分布式方案中对应的转发机制和分布式算法。

2.1 转发机制

对于任意目的节点，每个节点在转发表中维护两个下一跳，分别是最优下一跳和备份下一跳。下面描述节点选择下一跳的方法，当某个节点收到报文时，分为 3 种情况：

(1)若该节点到目的节点的最优下一跳没有故障，则将报文转发给最优下一跳；

(2)若该节点到目的节点的最优下一跳出现故障，则将报文转发给备份下一跳；

(3)该节点从最优下一跳接收到报文，由于该节点到目的节点的最优路径必定出现了故障，因此该节点需要将报文转发给备份下一跳。

2.2 分布式算法

下面详细描述分布式计算方案，不失一般性，假设目的地址为 d，源地址为 a。在链路状态路由协议中，每个节点都拥有该自治系统内的整个拓扑结构，因此节点 a 可以计算以目的地址 d 为根的最短路径树，假设 a 到 d 的最短路径是 $(a, m_1, m_2, m_3, \cdots, m_k, d)$，在分布式算法中，只需要计算该路径中节点的备份下一跳即可，而没有必要计算网络中所有节点的备份下一跳。例如在图 2 中，如果节点运行保护算法，则只需要首先计算出节点 (a,c,e) 对应的备份下一跳，然后为节点 f 计算备份下一跳，当计算出 f 对应的备份下一跳后程序执行完毕。

由以上描述可以看出，分布式算法只需要对集中式算法做微小的改动。对于某个节点，当计算出该节点的备份下一跳时，算法将执行完毕，而没有必要为其他节点计算备份下一跳，因此将大大减少算法的执行时间。并且在计算出该节点的备份下一跳之前，该节点不需要为已经计算出的其他节点下一跳存储备份下一跳，这将大大缩小算法占据的存储空间。

3 实验及结果分析

本节将通过实验来测试不同算法的运行结果。为了运行算法，我们需要获得网络拓扑结构。为了更加准确地评估算法的结果，将算法分别在不同类型的拓扑结构中运行，这些拓扑结构包括美国教育网络 Abilene[22] 和使用 Rocketfuel[23] 测量出的真实网络拓扑结构，这些拓扑结构都可以从互联网中获得，利用波士顿大学的开源项目提供的软件 Brite[24] 产生拓扑结构。

(1)Abilene 是美国教育网的拓扑结构，该拓扑可以从互联网中获得，该网络由 11 个路由器和 14 条链路构成。

(2) Rocketfuel 项目公布了大量的拓扑结构，我们在其中筛选了 4 个作为本实验的研究对象，它们的节点数量、链路数量、AS 名称等信息在表 1 中做了详细说明。

表 1　Rocketfuel 拓扑参数

AS 号码	AS 名称	节点数量	链路数量
3967	Exodus	79	147
1221	Telstra	108	153
3257	Tiscali	161	328
1239	Sprint	315	972

(3)Brite 是一个开源的软件，有 C 语言和 JAVA 语言两种版本。我们设置其 model 为“Waxman”，mode 的值为“Router only”，拓扑对应的节点数量分布在 20～1000 区间，节点的度分布在 2～40 区间，并且网络中链路的代价是对称的[25]，其值为带宽的倒数。Brite 软件对应的参数在表 2 中做了详细说明。

表 2　Brite 生成拓扑结构的参数设置

参数	值	参数	值
Model	Waxman	GrowthType	Incremental
N	20～1000	alpha	0.15
HS	1000	beta	0.2
LS	100	BWDist	Constant
m	2～40	BwMin-BwMax	10.0～1024.0
NodePlacement	Random	model	Router-only

为了有效说明所提算法的性能，在实验中比较了 Link-protection，LFA 和 U-turn 的实验结果。利用 C++语言实现了上述 3 种算法，并在 PC 机上模拟了实验。PC 机配置如下：CPU 采用英特尔芯片 inter i5，运行内存为 1.5GB，主频为 1.7GHz。

3.1 理论性分析

为了说明算法的高效性，从理论上分析算法的性能。表 3 列出了不同算法对应的数据，其中 k 表示节点邻居个数，m 表示节点邻居与这些邻居对应的邻居的个数总和。从表 3 可以看出，Link-protection 可以保护网络中所有单链路故障，但 LFA 和 U-turn 只能保护部分单链路故障。如果采用集中式计算方案，则 U-turn 的复杂度最高，Link-protection 的复杂度最低。如果采用分布式方案，则 LFA 和 U-turn 具有同样的复杂度，Link-protection 的复杂度最低。因此无论采用集中式还是分布式方案，Link-protection 都低于其余两种算法的复杂度。

表 3　算法理论性能比较

	Link-protection	LFA	U-turn
单链路故障	Y	N	N
算法复杂度(分布式)	$\|V\|O(\|V\|+\|E\|)$	$kO(\|V\|^2)$	$mO(\|V\|^2)$
算法复杂度(集中式)	$\|V\|O(\|V\|+\|E\|)$	$O(\|V\|^3)$	$O(\|V\|^3)$

3.2 故障保护率

本节评价不同方案对网络中单链路故障的保护能力。用故障保护率来衡量算法的保护能力。将故障保护率定义为：当网络中出现故障时，网络中所有连通的源-目的对的数量与所有源-目的对数量的比值。因此，算法的故障保护率越高，性能越好。

表 4 列出了不同算法在真实拓扑和利用 Rocketfuel 推测的网路拓扑的故障保护率。从表 4 可以看出，算法 Link-protection 可以保护网络中所有的单链路故障，明显优于 U-turn 和 LFA，该数据符合理论结果。

表 4　不同算法对应的单链路故障保护率/%

网络拓扑	Link-protection	LFA	U-turn
Abilene	100	54.28	84.67
Exodus	100	63.28	91.23
Telstra	100	53.46	80.85
Tiscali	100	75.84	90.13
Sprint	100	79.87	94.24

图 3 反映了故障保护率随着节点平均度的变化情况。从图 3 可以看出，随着节点平均度的增加，LFA 和 U-turn 的故障保护率有所提高，然而当平均度为 30 时，故障保护率依然低于 85%，Link-protection 可以提供 100%的单链路故障保护率。

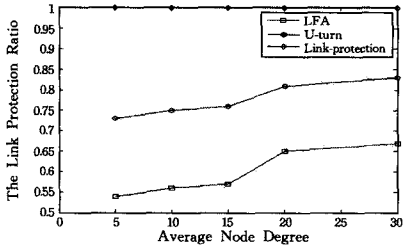

图 3 故障保护率随着节点平均度的变化情况

3.3 增量部署

本文提出的算法 Link-protection 和互联网部署的链路状态路由协议是兼容的，因此该算法支持增量部署。增量部署问题可以描述为：给定某个网络拓扑结构和需要部署节点的数量，目标是如何选择一组节点来部署该算法，从而使得故障保护率最高。下面实验采用贪心算法进行增量部署，即每次选择一个使得故障保护率提高最多的节点进行部署。

图 4 描述了算法 Link-protection，LFA 和 U-turn 在 Sprint 拓扑执行上述的增量部署方案时故障保护率随部署节点数量的变化情况。实验结果表明，随着部署节点数量的增加，故障保护率增加。当部署 50%左右的节点时，网络故障保护率有了大幅度的提高。当部署同样数量的节点时，Link-protection 的故障保护率明显优于 LFA 和 U-turn。

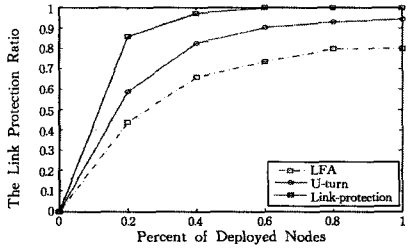

图 4 故障保护率随着部署节点数量的变化情况

3.4 路径拉伸度

本节评价当网络发生单链路故障后不同算法的路径拉伸度。路径拉伸度定义为：当网络出现故障后，使用重路由算法的路径长度与新的最短路径长度的比值。因此，路径拉伸度越小，重路由路径越接近于最短路径。本文采用的实验方法如下：随机断开网络中的某条链路，分别计算 3 种算法对应的重路由路径，从而得到路径拉伸度。重复上述实验 50 次，计算平均值。

图 5 描述了不同算法对应的路径拉伸度。从图 5 可以看出，LFA 和 U-turn 方案的路径拉伸度明显高于算法 Link-protection，这是因为 LFA 和 U-turn 采用从备份下一跳随机选择的方式，不能保证使用较短的重路由路径。算法 Link-protection 保护的路径拉伸度较小，因为在计算时该算法选择代价最小的路径作为重路由路径。

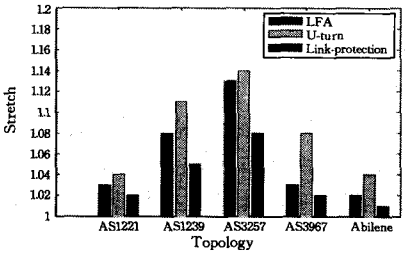

图 5 路由拉伸度

结束语 本文深入研究了网络中域内路由协议收敛过程中出现的路由环路和报文丢失等问题，在此基础上介绍了一种逐跳转发方式的路由保护方案。该路由保护方案可以应对网络中所有可能出现的单链路故障情形，并且其对应的报文的转发方式和互联网实际运行的路由协议的转发方式是一致的。因此，该方案很容易在互联网中实现。理论分析和实验结果表明，该方案可以 100%保护网络中所有单链路故障。网络中单节点和并发故障的发生率为 30%，然而上述算法并不能很好地应对这些故障。因此，下一步将重点研究单节点和并发故障对应的全保护方案。

参 考 文 献

[1] LEE S, YU Y, NELAKUDITI S, et al. Proactive vs Reactive Approaches to Failure Resilient Routing[C]// INFOCOM. 2004: 1-9.

[2] KRIST P. Scalable and Efficient Multipath Routing: Complexity and Algorithms[C]// 2015 IEEE 23rd International Conference on Network Protocols (ICNP). IEEE, 2015: 376-385.

[3] NUCCI A, BHATTACHARYYA S, TAFT N, et al. IGP Link Weight Assignment for Operational Tier-1 Backbones [J]. IEEE/ACM Transactions on Networking (TON), 2007, 15(4): 789-802.

[4] MOTIWALA M, ELMORE M, FEAMSTER N, et al. Path splicing[C]// SIGCOMM. 2008: 27-38.

[5] MOTIWALA M, FEAMSTER N, VEMPALA S. Path splicing: Reliable connectivity with rapid recovery[C]// HotNets. 2007: 1-7.

[6] ERLEBACH T, MEREU A. Path splicing with guaranteed fault tolerance[C]// GLOBECOM. 2009: 1-6.

[7] CHO S, ELHOURANI T, RAMASUBRAMANIAN S. Independent directed acyclic graphs for resilient multipath routing [J]. IEEE/ACM Transactions on Networking (TON), 2012, 20(1): 153-162.

[8] LAKSHMINARAYANAN K, CAESAR M, RANGAN M, et al. Achieving convergence-free routing using failure-carrying packets [C]// SIGCOMM. 2007: 241-252.

[9] YABG B, LIU J, SHENKER S, et al. Keep forwarding: Towards k-link failure resilient routing [C] // INFOCOM. 2014: 1617-1625.

[10] KWONG K, GAO L, GUERIN R, et al. On the feasibility and efficacy of protection routing in IP networks [J]. IEEE/ACM Transactions on Networking (ToN), 2011, 19(2): 1543-1556.

[11] YUAN Q, WU J, CARDEI I. Smrs: A scalable multi-path routing scheme[J]. International Journal of Parallel, Emergent and Distributed Systems, 2009, 24(1): 69-84.

[12] ENYEDI G, RÉTVÁRI G, SZILÁGYI P, et al. IP Fast Re-

Route: Lightweight Not-Via without Additional Addresses[C]// INFOCOM. 2009: 157-168.

[13] GUMMADI, KRISHNA P, et al. Improving the reliability of internet paths with one-hop source routing[C]// Conference on Symposium on Opearting Systems Design & Implementation USENIX Association. 2004: 13-13.

[14] FORTZ B, THORUP M. Optimizing OSPF/IS-IS Weights in a Changing World [J]. IEEE Journal on Selected Areas in Communications, 2006, 20(4): 756-767.

[15] ATLAS A, ZININ A. Basic specification for ip fast reroute: Loop-free alternates[C]// RFC 5286. 2008: 1-32.

[16] YANG X, WETHERALL D. Source selectable path diversity via routing deflections[C]// SIGCOMM. 2006: 159-170.

[17] ATLAS A. U-turn A lternates for Ip/LDp Fast-Reroute[C]// IETF Internet Draft. 2006: 1-29.

[18] SRIDHARAN A, GUERIN R, DIOT C. Achieving near-optimal traffic engineering solutions for current ospf/is-is networks[J]. IEEE/ACM Transactions on Networking (TON), 2005, 13(2): 234-247.

[19] FRANCOIS P, BONAVENTURE O. An evaluation of ip-based fast reroute techniques[C]// CoNEXT. 2005: 244-245.

[20] RETVÁRI G, CSIKOR L, TAPOLCAI J, et al. Optimizing igp link costs for improving ip-level resilience[C]// Design of Reliable Communication Networks (DRCN). 2011: 62-69.

[21] MARKOPOULOU A, IANNACCONE G, BHATTACHARYYA S, et al. Characterization of Failures in an Operational IP Backbone Network [J]. IEEE/ACM Transactions on Networking (TON), 2008, 16(4): 749-762.

[22] NETWORK A B. Advanced networking for research and education[OL]. http://abilene.internet2.edu, 2003.

[23] SPRING N, MAHAJAN R, WETHERALL D, et al. Measuring isp topologies with rocketfuel [J]. IEEE/ACM Transactions on Networking, 2004, 12(1): 2-16.

[24] MEDINA A, LAKHINA A, MATTA I, et al. Brite: An approach to universal topology generation [C]// IEEE International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems. 2001: 346-353.

[25] GJOKA M, RAM V, YANG X. Evaluation of ip fast reroute proposals[C]// Communication Systems Software and Middleware. 2007: 1-8.

（上接第 41 页）

表 2　机器故障检测和修复库所和变迁说明

节点	节点含义
$p15$	显示机器人状态
$p16$	传感器输入接口
$p17$	传感器输出接口
$p18$	机器人输入接口
$p19$	机器人输出接口
$p20$	故障输入接口
$p21$	标准参数输入接口
$p22$	显示故障原因
$p23$	机器人当前状态
$p24$	机器人重置后状态
$p25$	机器人 fault 状态
$t12$	检查机器人状态
$t13$	保存当前状态信息
$t14$	收集数据
$t15$	调校数据
$t16$	保存当前状态信息
$t17$	接收故障信息
$t18$	分析故障原因
$t19$	参数校正
$t20$	对机器人进行重置
$t21$	检查机器人状态

结束语　CPS 作为时空多维的安全关键系统，如何对其时空属性和非功能属性进行建模成为一大挑战。本文提出了一种面向方面的时空 Petri 网建模方法，该方法在一定程度上降低了系统的耦合性，提高了可扩展性和可维护性。以后工作将会对系统中的方面网进行形式化分析，并采用模型检测工具对系统关键属性进行分析和验证。

参 考 文 献

[1] HE J F. Cyber-Physical Systems[J]. China Computer Federation, 2010, 6(1): 25-29. (in Chinese)
何积丰. Cyber-Physical Systems[J]. 中国计算机学会通讯, 2010, 6(1): 25-29.

[2] GIRAULT C, VALK R. Petri Nets for Systems Engineering: A Guide to Modeling, Verification, and Applications [M]. Springer & Business Media, 2013.

[3] ZHANG M T, ZHANG G Q, ZHANG K, et al. Spatio-Temporal Petri Net Model for Describing CPS Physical Entities[J]. Journal of Frontiers of Computer Science & Technology, 2013, 7(4): 377-384. (in Chinese)
张茗泰，张广泉，张侃，等. 描述 CPS 物理实体的时空 Petri 网模型[J]. 计算机科学与探索，2013，7(4)：377-384.

[4] KICZALES G, LAMPING J, MENDHEKAR A, et al. Aspect-oriented programming[C]// European Conference on Object-oriented Programming. Springer Berlin Heidelberg, 1997: 220-242.

[5] ROUBTSOVA E, AKSIT M. Extension of Petri Nets by Aspects to Apply the Model Driven Architecture Approach [C]// Preliminary Proceedings of the 1st International workshop on Aspect-Based and Model-Based Separation of Concerns in Software Systems (ABMB). 2005: 1-15.

[6] XU D, NYGARD K E. Threat-Driven Modeling and Verification of Secure Software using Aspect-Oriented Petri Nets[J]. IEEE Transactions on Software Engineering, 2006, 32(4): 265-278.

[7] QIAN Z, YU H. A TAOPN Approach to Modeling and Scheduling Cyber-Physical Systems[C]// International Conference on Information Science and Applications (ICISA). IEEE, 2013: 1-7.

[8] ZHENG L, RONG M, Zhang G, et al. Method for Modeling CPS Based on Open Colored Petri Net [J]. Journal of Computational Information Systems, 2014, 10(22): 9657-9664.

[9] JENSEN K. An Introduction to the Theoretical Aspects of Coloured Petri Nets [M]// A Decade of Concurrency Reflections and Perspectives. Springer Berlin Heidelberg, 1994: 230-272.

[10] CHRISTENSEN S, HANSEN N D. Coloured Petri Nets Extended with Place Capacities, Test Arcs and Inhibitor Arcs[M]. Springer Berlin Heidelberg, 1993.