Computer Communications 35 (2012) 2082–2092

Contents lists available at SciVerse ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Minimum Protection Cost Tree: A tunnel-based IP Fast Reroute Scheme

Mingwei Xu [a,*], Qing Li [a], Lingtao Pan [a], Qi Li [a], Dan Wang [b]

[a] *Department of Computer Science and Technology, Tsinghua University, Beijing, PR China*
[b] *Department of Computing, The Hong Kong Polytechnical University, Hong Kong*

ARTICLE INFO

Article history:
Received 26 June 2011
Received in revised form 14 June 2012
Accepted 15 June 2012
Available online 29 June 2012

Keywords:
IP Fast Reroute
Routing Resilience
MPCT
Tunnel

ABSTRACT

Diverse delay-sensitive realtime applications, e.g., Voice over IP, are emerging in the Internet and bringing a rigorous requirement on the transmission delay. Therefore, several IP Fast Reroute (IPFRR) schemes have been proposed. However, these schemes are all either too computationally expensive or unsatisfactory in protection coverage. In this paper, we propose Minimum Protection Cost Tree (MPCT), a tunnel-based IP Fast Reroute Scheme. First, MPCT provides 100% single-node protection coverage with direct forwarding (DF). Second, the computational complexity of MPCT is less than one full shortest path first (SPF) calculation. Third, by simulation with the data from China Education and Research Network (CERNET), Rocketfuel and BRITE, we show that even without DF, MPCT can provide more than 99% protection coverage. Besides, the protection path stretch ratios in real topologies are between 10% and 20%, which is acceptable compared with the corresponding achievement of reducing the algorithm complexity. We believe that our scheme MPCT moves a big step towards practical deployment.

1. Introduction

The current Internet routing protocols take on the order of a few hundred milliseconds or even tens of minutes to re-converge after failure [1,2]. However, recent popularity of online realtime applications, such as Voice over IP (VoIP), streaming media, and telecommuting/video, has led to a rigorous requirement on the transmission delay of the Internet [3]. Internet Service Providers (ISPs) hence have strong incentives to improve the network survivability during failures.

There are two main approaches to improve network survivability, namely protection and restoration [4–6]. *Protection* is based on pre-computed backup paths: a protection (backup) path is pre-computed to forward the traffic when the primary path fails. *Restoration* computes a new path after a failure has occurred. In this paper, we focus on the approach of protection, which has proved to be efficient in shortening the service disruption time caused by failures.

IP fast reroute (IPFRR) [7–12] is one of the most significant branches of routing protection. In IPFRR, the failure-adjacent nodes (protection source nodes) pre-compute backup paths, which can be used to protect failure-affected packets immediately upon the detection of failures. Traffic can be delivered to the correct destinations even before the network re-converges on the new topology. For practical deployment, an IPFRR scheme should (1) provide high protection coverage and (2) introduce slight overhead (e.g., computation and memory) in the current routing protocol.

There are several intra-domain IPFRR mechanisms: loop free alternates (LFA) [7], Efficient Scan for Alternate Paths (ESCAP) [8], NotVia [9] and IP Fast Reroute using tunnels (Tunnel) [10], none of which can satisfy the above conditions. Each of these schemes is either unsatisfactory in protection coverage or too computationally expensive. Although LFA has low complexity, it only provides about 40% single-node protection coverage [13]. While ESCAP and NotVia can provide 100% protection coverage, they involve nontrivial computational and memory overhead. Tunnel can provide more than 90% single-node protection coverage (better than LFA). Besides, Tunnel involves little memory overhead, compared with ESCAP and NotVia. However, Tunnel needs $|\mathbf{V}| - 1$ ($\mathbf{V}$ is the node set of the intra-domain topology) times of shortest path first (SPF) calculations to find the available tunnel end points. The computation complexity blocks Tunnel from practical deployment.

In this paper, we propose the Minimum Protection Cost Tree (MPCT), which is based on IP-in-IP tunnel as NotVia and Tunnel. The development of the algorithm MPCT can be divided into two main steps. First, based on the incremental SPF (iSPF) [14], we introduce Tunnel-AT, a naive algorithm to compute the tunnel end points. We prove Tunnel-AT can provide 100% protection coverage for single-node failures by re-protection (re-protecting a packet after decapsulation) and direct forwarding. Direct forwarding is an extension of IP-in-IP tunnel, which directs the tunnel end point to forward the packet to a specified neighbor after decapsulation [10]. Currently, it may be not well accepted by the ISPs. Thus,

* Corresponding author. Tel.: +86 010 62603057; fax: +86 010 62603064.
E-mail addresses: xmw@cernet.edu.cn (M. Xu), liqing@csnet1.cs.tsinghua.edu.cn (Q. Li), plt@csnet1.cs.tsinghua.edu.cn (L. Pan), liqi@csnet1.cs.tsinghua.edu.cn (Q. Li), csdwang@ployu.edu.hk (D. Wang).

0140-3664/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.comcom.2012.06.010

in order to avoid unnecessary direct forwarding in Tunnel-AT as far as possible, we provide Minimum Protection Cost Tree (MPCT) to compute the protection paths. In MPCT, the protection path without direct forwarding is prior to that with direct forwarding; the protection path without re-protection is prior to that with re-protection. This priority guarantees the protection paths with the minimized protection costs are selected. In this way, MPCT can provide higher protection coverage even without the support of direct forwarding from ISPs.

We demonstrate the performance of MPCT by comprehensive simulation based on three groups of topologies: the topology of Chinese Education and Research Network (CERNET) [15], the measured topologies from Rocketfuel project [16] and the diverse topologies generated by BRITE [17]. The simulation results show:

- The computation overhead introduced by MPCT is much less than LFA, tunnel and NotVia. MPCT consumes less than one time of full shortest path first (SPF) calculation, which guarantees that MPCT will not become the new computational bottleneck in the intra-domain routing protocol OSPF.
- The protection coverage of MPCT is 100% with direct forwarding, compared with 40% of LFA, 90% of Tunnel and 100% of NotVia. Besides, MPCT can provide more than 90% protection coverage for multiple concurrent failures (with six failed links in our simulation).
- The ratios of direct forwarding and re-protection are both less than 1% in MPCT. This means even without the support of direct forwarding from ISPs, MPCT can provide more than 99% single-node protection coverage. Besides, the overhead caused by re-protection can be nearly ignored.
- The protection path stretch ratios of MPCT range from 10% to 20% in Rocketfuel topologies, which is acceptable compared with the achievement of reducing the computation overhead.

The rest of the paper is organized as follows: Section 2 contains the background of our work. We propose the scheme of MPCT in Section 3 and provide the corresponding algorithm in Section 4. Simulation results are provided in Section 5. Section 6 is devoted to existing solutions and comparison. Section 7 concludes the paper.

2. Background

In this section, we discuss Tunnel [10] and incremental shortest path first (iSPF) algorithm [14], which are the foundation of our work.

2.1. IP Fast Reroute using tunnel

With the popularity of realtime applications, e.g., VoIP, it is becoming more and more imperative to improve the network performance during failures, especially the performance of transmission delay. Therefore, many schemes are proposed to provide backup routes before the network re-converges. Among these schemes, IP Fast Reroute using tunnels [10] (Tunnel for short) is one of the most significant. Compared with LFA [7], Tunnel provides higher protection coverage; compared with ESCAP [8] and NotVia [9], Tunnel involves little memory overhead. However, Tunnel also has its own problems.

Before diving into the details of Tunnel, we first give some notations. Let $\mathbf{G}(\mathbf{V},\mathbf{E})$ be the default network topology in this paper, where $\mathbf{V}$ is the node set and $\mathbf{E}$ is the link set. Let $s \in \mathbf{V}$ be the default protection source node (the failure-adjacent router) that detects the failure and starts the protection path. Let $d \in V$ be the default failure-affected destination node. Let $f \in \mathbf{V}$ be the default failed node. We can see that f is the next hop from s to d. The notations used in the paper are summarized in Table 1.

In Tunnel, s pre-computes an available tunnel end point (TEP) t that satisfies the following condition: neither of the optimal routes from s to t and from t to d is via f. As shown in Fig. 1(a), if f fails, t is an available tunnel end point from s to d. Thus, s can encapsulate the packet (originally to d) with the address of t; t will receive the packet and decapsulate it. According to the above condition, the protection path from s to t to d is safe, because it is not affected by the failure.

Tunnel fails if no available tunnel end point exists. An example can be found in Fig. 1(b). Direct forwarding (DF) is introduced to solve the problem. DF is an enhancement of IP-in-IP tunnel to direct the tunnel end point to forward the packet to the specific neighbor. Directed forwarding might be implemented by an enhancement to the IP tunneling encapsulation or a single interposed MPLS label stack entry [10]. As Fig. 1(b) shows, p can be selected as the tunnel end point and q is the corresponding DF neighbor. However, even Tunnel with DF might fail, for which Fig. 1(c) shows an example.

Actually, the main obstacle that blocks Tunnel from practical deployment is the lack of efficient algorithm. To find the protection routes for all possible failures, s has to compute the (reverse) shortest path trees (SPTs) rooted all other nodes in the network, which involves $|\mathbf{V}| - 1$ times of Short Path First (SPF) calculation and brings a new bottleneck for route computation.

2.2. Incremental shortest path first algorithm

Incremental shortest path first (iSPF) algorithm [14] is proposed to recalculate the new shortest path tree (SPT) when the network topology changes by one single node or link. In the following section, we will propose our algorithm of computing the protection paths for the failure-affected destinations based on iSPF algorithm.

Intuitively, the new SPT does not change too much from the old one when only a link or node changes. Thus, currently adopted full SPF recalculation in OSPF is inefficient. Based on this observation, iSPF performs as follows: (1) only those affected nodes in the SPT will be adjusted; (2) in every iteration, a subtree, instead of only one node, is appended to the new SPT. We use Fig. 2 to illustrate the algorithm. Fig. 2(a) contains six nodes with labels from A to F. Solid lines are links in the shortest path tree rooted at node A. Dotted lines are links not in the tree. The weight of each link is shown beside the link. The number in a node is the distance from A to that node.

Table 1
The notations used for scheme description.

Notation	Definition
$\mathbf{G}(\mathbf{V},\mathbf{E})$	The default network topology in the paper
s	The default protection source node
d	The default failure-affected destination
f	The default failed node
$dist(x,y)$	The optimal dist from x to y
$dist'(x,y)$	The dist from x to y in the new SPT after f fails
$\widehat{dist}(x,y)$	The dist from x to y in the MPCT after f fails

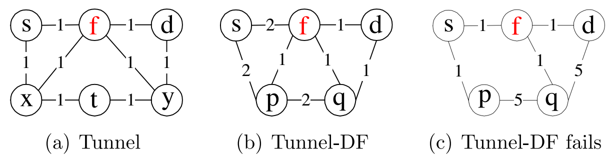

Fig. 1. (a) An example of an available tunnel end point. (b) An example of tunnel with direct forwarding. (c) An example of tunnel with direct forwarding failing.

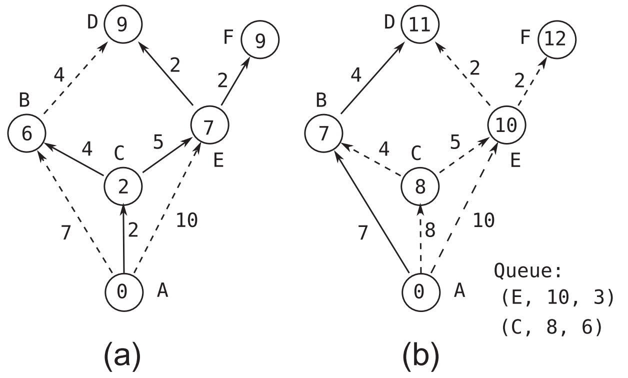

Fig. 2. The incremental shortest path first algorithm.

As shown in Fig. 2(a), the cost of link (A, C) changes from 2 to 8. Nodes from B to F are all *affected nodes* and are marked as *floating*. Only the root node A is marked as *attached*. For all the floating nodes, we check whether they have links to the attached nodes, and calculate the new distance and the change (δ) in distance, which gives queue $\{(B, 7, 1), (E, 10, 3), (C, 8, 6)\}$, where the third value is the δ. In the next iteration, $(B, 7, 1)$ will be considered first since it has the smallest delta. B will be marked as attached. Since B has an outing edge to D, the new distance and δ of D is calculated, which gives queue $\{(D, 11, 2), (E, 10, 3), (C, 8, 6)\}$. Then $(D, 11, 2)$ is selected and D is marked as attached. We show the current result in Fig. 2(b).

In the next iteration, node $(E, 10, 3)$ is selected, instead of only marking E as attached, *the original subtree rooted at E is considered together*. Both E and F are marked as attached, and the new optimal dist of F is 12, which is calculated by adding the δ ***(3) to its original dist. In the next iteration, node C is selected and the new shortest path tree is computed.

3. Minimum Protection Cost Tree: methodology

In this section, we propose our scheme MPCT. First, we propose a naive solution (Tunnel-AT) based on iSPF, which we prove can provide 100% protection coverage. However, Tunnel-AT involves some unnecessary direct forwarding. Therefore, we introduce our scheme, Minimum Protection Cost Tree, which avoids unnecessary direct forwarding. Before diving into the details of the methodology, we first make some explanations on special cases in our scheme.

Current IPFRR schemes, including LFA, NotVia and Tunnel, all focus on single-node (or single-link) failures. We follow this tradition to focus on single-node failures. When the source router s finds that the optimal next hop f to the destination d is unreachable, s assumes that f fails and triggers the pre-computed backup route. However, the destination itself might be the optimal next hop (*the last-hop problem*). To cover this special case, we always suppose that all out-links of f (the links with f as the tail) and the link from s to f fail. In this way, the general case (s is not d) and the special case (f is d) are both covered. In what follows, to simplify the scheme design, we use the general case to discuss our scheme. The design principles and algorithms can be applied in both cases.

As aforementioned, we generally concentrate on designing a solution for single-node failures in this paper. However, potential multiple simultaneous failures might lead to routing loops, which would consume network bandwidth. Therefore, we provide two approaches to handle this problem:

1. The packet addressed to the tunnel end point will be dropped when encountering another failure. Therefore, the protected packet can only be re-protected after decapsulation.
2. As suggested in [9], a router can record the list of the nodes the protection path traverses when computing the protection routes. When the router intends to protect an encapsulated packet, it must ensure that the original source node is not in the node list of the protection path, which guarantees no loop.

The two approaches both have their own pros and cons. Compared with the first, the second introduces more memory overhead for the protection of interactional simultaneous failures. Besides, actually 86.5% network failures are single-link or single-node failures [18], therefore, we choose the first method in our paper. Further study is required on this point, which is outside the scope of this document.

3.1. A naive solution Tunnel-AT

We first propose a naive solution Tunnel-AT [19] based on iSPF. In each iteration of iSPF: (1) the node with the smallest change of dist is selected; 2) a subtree, instead of only one node, is added to the new SPT. The process of iSPF can be viewed as attaching subtrees back one by one until the new SPT forms. We first introduce some definitions before diving into the details of Tunnel-AT.

An *attaching tree* (*ATTree* for short) is a subtree attached to the new SPT in an iteration. For example, in Fig. 3, Fig. 3(a) is an example of topology; Fig. 3(b) is the corresponding SPT rooted at s; Fig. 3(c) shows the steps of iSPF to form a new SPT after f fails. $\{d_1\}$ is the *ATTree* of the first iteration; $\{d_2, d_3\}$ and $\{d_4\}$ are respectively the second and third. Connected *ATTree*s in the new SPT form a *super ATTree*. For example, in Fig. 3, $\{d_1\}$ and $\{d_2, d_3, d_4\}$ are *super ATTrees*. We can see each failure-affected node is in one of the *super ATTrees*. The *incoming node* for any failure-affected destination is the root of the corresponding *super ATTree*. The *attaching node* for any failure-affected destination is the father of the corresponding *incoming node* in the new SPT. For example, in Fig. 3, d_2 and p are respectively the *incoming node* and *attaching node* for d_2, d_3 and d_4.

Tunnel-AT Scheme. When f fails, s reroutes the packet to the corresponding *attaching node* by tunnel, or to the corresponding *incoming node* by tunnel with direct forwarding if f is on the route from the *attaching node* to the destination. For example, in Fig. 3, if f fails, all the packets to d_2, d_3 and d_4 will be sent to d_2 by tunnel with direct forwarding. While the packet to d_1 will be sent to p by tunnel.

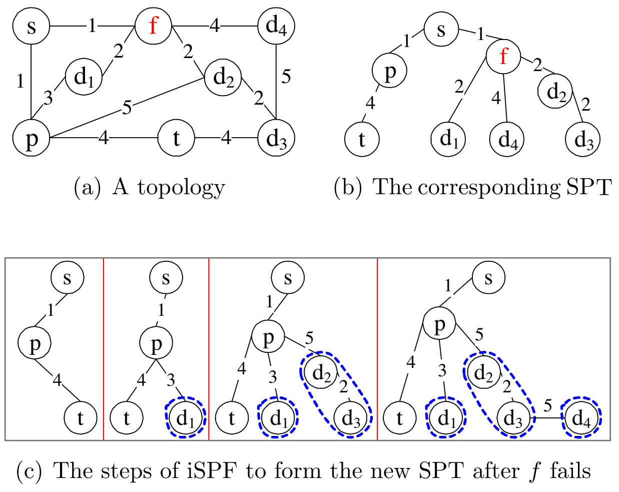
(a) A topology
(b) The corresponding SPT
(c) The steps of iSPF to form the new SPT after f fails

Fig. 3. (a) An example of topology. (b) The corresponding s-rooted SPT of (a). (c) The steps of iSPF to form the new SPT after fails. $\{r\}$, $\{q, h\}$ and $\{d\}$ are *ATTrees*. $\{r\}$ and $\{q, h, d\}$ are *super ATTrees*.

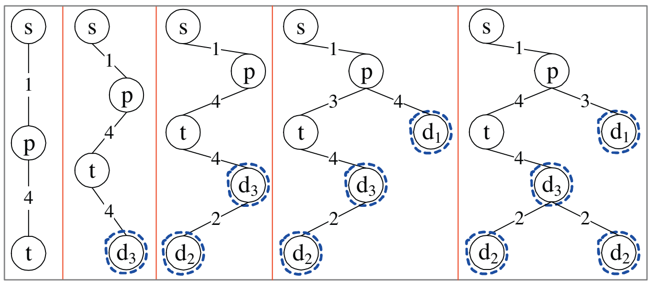

Fig. 4. MPCT construction with the topology in Fig. 3(a) and the SPT in Fig. 3(b) as the input. s and f are still the protection source node and the failed node respectively.

Table 2
Notations used in Tunnel-AT routing algorithm.

Notation	Definition
$\mathcal{T}$	The shortest path tree rooted at s
$I(x), I(N)$	In edges of node x or of a set of nodes N
$O(x), O(N)$	Out edges of node x or of a set of nodes N
$S(e)$	The source node of edge e
$E(e)$	The end node of edge e
Q	A priority queue
$\{d, (p, dist, \omega)\}$	An item in Q : d is a destination, p is its potential parent, $dist$ is a potential distance, ω is a potential Ω.
$ATTree(r)$	The ATTree rooted at r in the MPCT.
D	The set of all affected nodes.

Table 3
The characteristics of Rocketfuel AS topologies.

AS number	Name	#Routers	#Links	Average degree
1221	Telstra (au)	104	151	2.90
1239	Sprint (us)	315	972	6.17
1755	Ebone (eu)	87	161	3.70
3257	Tiscali (eu)	161	328	4.07
3967	Exodus (us)	79	147	3.72
6461	Abovenet (us)	128	372	5.81

Table 4
Parameters of BRITE topologies.

Mode	Model	HS	LS	Nodes num
Router only	Waxman	1000	100	50–500
Links/new node	α/β	NP	Growth type	
2–14	0.15/ 0.2	Random	Incremental	

In what follows, we prove that Tunnel-AT can provide 100% protection coverage for single-node failures.

Lemma 1. *All the destination nodes in a super ATTree can be protected by rerouting the packets to the corresponding incoming node.*

Proof. Let d be any node in the *super ATTree*. Let q be the *incoming node*. Let p be the *attaching node*. If the optimal route from (q, d) is not via f, the packet is safe after being sent to q, and no problem exists; otherwise, the packet is still not safe. Now we prove that the packet will be delivered to d even if f is on the route from q to d.

We can see that s is not on the optimal route from q to f, because f is a node on the optimal route from s to q. Thus, before reaching f, the protected packet will be received by another protection source node s_1. s_1 will start another protection action (*re-protection*) to reroute the packet to a new *incoming node* from the view of s_1. Let $dist(x, y)$ be the optimal dist from x to y in the topology $\mathbf{G}$. Let $dist'(x, y)$ be the optimal dist from x to y in the new topology without f. We have

$$\begin{aligned} dist'(s_1, d) &\leqslant dist'(s_1, q) + dist'(q, d) \\ &= dist(s_1, q) + dist'(q, d) \\ &< dist(s, q) + dist'(q, d) \\ &\leqslant dist(s, p) + linkcost(p, q) + dist'(q, d) \\ &= dist'(s, d) \end{aligned}$$

Thus, s_1 is closer to d than s in the new topology without f. This monotonicity means that after each protection action, the packet moves closer towards the destination. Thus, the packet will finally reach the destination d. □

For example, in Fig. 3, after detecting f fails, s reroutes the packet (originally to d_4) to d_2 by tunnel with direct forwarding. If it is really f that fails, d_2 starts another protection action. d_2 encapsulates the packet to d_3. After being decapsulated by d_3, the packet becomes safe. If it is the link (s, f) that fails, no *re-protection* will occur.

Theorem 1. *Tunnel-AT can provide 100% single-node protection coverage.*

Proof. Each failure-affected destination node is in one *super ATTree*. Thus, according to Lemma 1, all the affected destinations can be protected by Tunnel-AT. □

We accept the overhead of *re-protection* because actually 70% of the failures are single-link failures [18]. Mostly, it is the *link*(s, f) not the node f that fails. In this case, no *re-protection* will be triggered and the protection path is optimized a lot. The same problem exists in ED-NotVia [20], where more detailed explanations about re-protection can be found.

Tunnel-AT is efficient in computing protection paths. However, the protection paths of Tunnel-AT are only optimal in the length but not in the cost. According to the simulation, more than 30% of the protection paths need direct forwarding in Tunnel-AT. Only a small part of the direct forwarding is indeed necessary. For example, in Fig. 3, for d_2, d_3 and d_4, direct forwarding is needed in Tunnel-AT. For d_4, *re-protection* is triggered. However, if t is selected as the tunnel end point, no direct forwarding and re-protection occur. Although direct forwarding and *re-protection* are acceptable, they are more costly compared with protection path length, and thus should be avoided as far as possible. Therefore, the length should not be the only parameter in protection path selection.

3.2. Minimum Protection Cost Tree

Currently, direct forwarding may not be well accepted by ISPs and re-protection causes extra overhead. Thus, protection paths relying on direct forwarding and re-protection should be assigned with the lowest priority. This is especially important for ISPs that do not support direct forwarding, because this guarantees a higher protection coverage even without direct forwarding. If the routers support direct forwarding, MPCT can provide 100% protection coverage; if not, MPCT can still achieve a higher protection coverage. In order to achieve this objective, we propose Minimum Protection Cost Tree (MPCT), where direct forwarding and re-protection are both transferred into costs and the protection paths with the minimized cost are preferred.

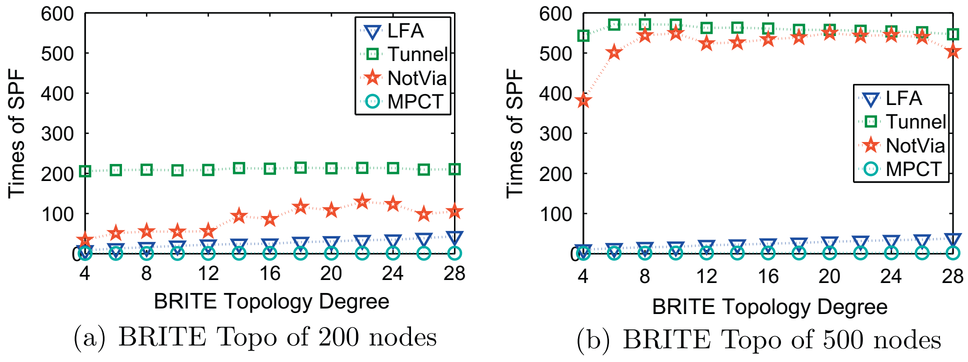

Fig. 5. Running times of IPFRR schemes on BRITE topologies. *X* axis is the BRITE topology degree and *Y* axis is (the running time of the IPFRR scheme)/(the running time of one SPF).

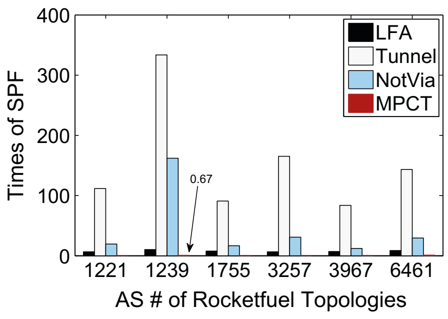

Fig. 6. Running times of IPFRR schemes on Rocketfuel topologies.

As an improvement over Tunnel-AT, MPCT instead of SPT is used for protection path computation. The concepts of *ATTree*, *super ATTree*, *incoming node* and *attaching node* still apply. Different from Tunnel-AT (or iSPF), in each iteration, the *ATTree* with the minimum protection cost will be attached. Let $\Omega = V_{DF} + V_{repro} + V_{path}$ be the protection cost of some ATTree. V_{DF} is associated with direct forwarding, V_{repro} is associated with re-protection, and V_{path} is associated with protection path length.

Now we provide the detailed definitions of V_{DF}, V_{repro} and V_{path} for a specific *ATTree*. Let $\mathcal{D}$ (*diameter*) be an constant that is larger than the longest path in the network. Let *q* be the potential *incoming node*. Let *p* be the potential *attaching node*. Let *r* be the root of the *ATTree*.

- V_{DF}. If *s* is not on the route from *p* to *r* (direct forwarding is not needed), $V_{DF} = 0$; or else, $V_{DF} = 2 \times \mathcal{D}$. Take the topology in Fig. 3(a) as an example (*f* fails), $\{r\}, \{q,h\}$ and $\{h\}$ are the potential *ATTrees* in the first iteration because they are floating and have unaffected neighbors. For $\{d_1\}$ and $\{d_3\}$, $V_{DF} = 0$; for $\{d_2, d_3\}$, $V_{DF} = 2 \times \mathcal{D}$.
- V_{repro}. (1) $V_{DF} = 0$. If *f* is on the route from *p* to *d* (re-protection is required), $V_{repro} = C_d$; or else, $V_{repro} = 0$. (2) $V_{DF} \neq 0$. If *f* is on the route from *q* to *d* (re-protection is required), $V_{repro} = C_d$; or else, $V_{repro} = 0$. For example, in Fig. 3(a), $V_{repro} = 0$ for all of $\{d_1\}$, $\{d_3\}$ and $\{d_2, d_3\}$ in the first iteration.
- V_{path}. $V_{path} = \widehat{dist}(p,d) - dist(s,p) - dist(s,d)$, where $\widehat{dist}(p,d)$ is the dist from *p* to *d* in the new tree to be formed. The smaller V_{path} is, the less possible that Direct Forwarding is required. In Fig. 3(a), for $\{d_1\}$, $V_{path} = -1$; for $\{d_2, d_3\}$, $V_{path} = 1$; for $\{d_3\}$, $V_{path} = -6$.

We show an example of MPCT construction with the topology in Fig. 3(a) and and the SPT in Fig. 3(b) as the input. *s* and *f* are still the protection source node and the failed node respectively. As shown in Fig. 4, in the first iteration, there are three potential *ATTrees*: $\{d_1\}$, $\{d_2, d_3\}$ and $\{d_3\}$. According to the above definitions, the protection costs for them are respectively -1, $1 + 2 \times \mathcal{D}$ and -6. Thus, the *ATTree* $\{d_3\}$ is attached below *t* in the first iteration. Then, $\{d_2, d_3\|$ is removed; $\{d_2\}$ and $\{d_4\}$ become new potential *ATTrees* (d_3 is the attaching node). The protection costs for $\{d_2\}$ and $\{d_4\}$ are respectively -2 and -1. Thus, the *ATTree* $\{d_2\}$ is attached below d_1 in the second iteration. Then $\{d_1\}$ and $\{d_4\}$ are attached in the following steps. The final MPCT shows the protection paths for all the failure-affected destinations. *p* is the selected tunnel end point for d_1; *t* is the selected tunnel end point for d_3, d_2 and d_4. In this example, direct forwarding and re-protection are completely avoided.

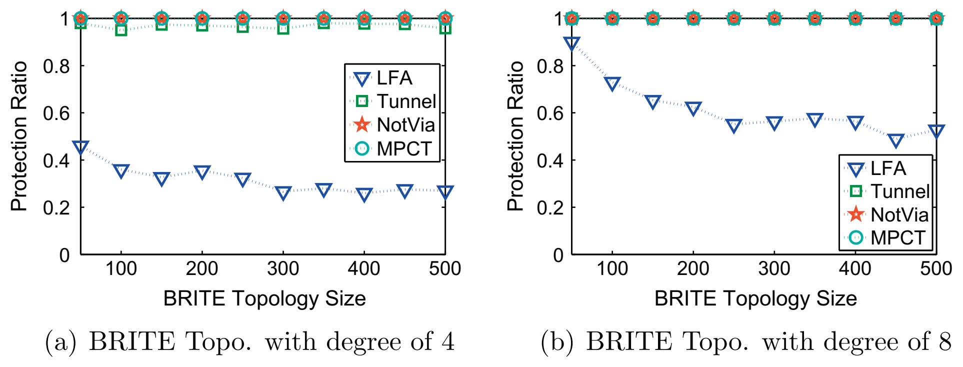

Fig. 7. Protection ratios of IPFRR schemes on BRITE topologies.

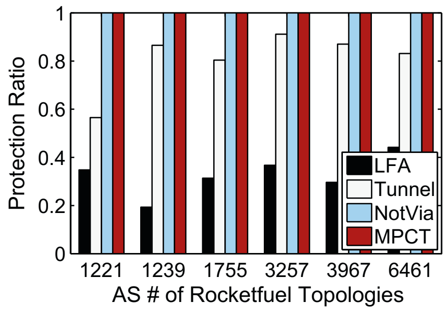

Fig. 8. Protection ratios of IPFRR schemes on Rocketfuel topologies.

MPCT Scheme. The forwarding behavior is the same as Tunnel-AT, except that the *incoming node* and the *attaching node* are in MPCT rather than the new SPT.

In the following sections of the paper, we use MPCT to denote both Minimum Protection Cost Tree and MPCT Scheme. MPCT provides 100% protection coverage for single-node failures, which can be proved similarly to Theorem 1. In the next section, we will propose the algorithm for MPCT construction and protection path selection.

4. Algorithm design for MPCT

Networks with asymmetric link weights would lead to a higher computational complexity. For simplicity, we focus on providing protections for the networks with the symmetric link weights and keep the asymmetric case as the future work.

4.1. Protection cost calculation

The challenge of designing an efficient algorithm is the calculation of protection cost for each *ATTree* in MPCT. Now we introduce the novel method for the protection source node s to calculate the protection cost, or the Ω value, for each *ATTree* of MPCT.

Before diving into the details of calculation, we first emphasize the fact that s can obtain $dist(s,x)$ and $\widehat{dist}(s,x)$ without any extra calculation. Let p be the *attaching node* and q be the *incoming node*.

First, $V_{path} = \widehat{dist}(p,d) - dist(s,p) - dist(s,d)$. $dist(s,p)$ and $dist(s,d)$ are known values. $\widehat{dist}(p,d)$ can be calculated according to $\widehat{dist}(p,d) = \widehat{dist}(s,d) - dist(s,p)$. Thus,

$$V_{path} = \widehat{dist}(s,d) - dist(s,p) - dist(s,p) - dist(s,d)$$

Second, if $V_{path} < 0, V_{DF} = 0$; or else $V_{DF} = 2 \times \mathcal{D}$. If $V_{path} < 0$, $dist(p,d) \leqslant \widehat{dist}(p,d) < dist(s,p) + dist(s,d)$, thus the packet encapsulated to p will not loop back to s after decapsulation and no direct forwarding is needed.

Now we discuss the calculation of V_{repro}. Let x be the end node of the protection action ($x = p$ if $V_{DF} = 0$; or else, $x = q$). If f is not on the route from x to d, no re-protection will be triggered, thus $V_{repro} = 0$. We use the following formula to calculate V_{repro}:

$$V_{repro} = \begin{cases} 0, & \widehat{dist}(x,d) < dist(x,f) + dist(f,d); \\ \mathcal{D}, & \text{else.} \end{cases}$$

We first prove the correctness of the above formula. As $dist(x,d) \leqslant \widehat{dist}(x,d)$ and $\widehat{dist}(x,d) < dist(x,f) + dist(f,d)$, $dist(x,d) < dist(x,f) + dist(f,d)$, thus f is not on $route(x,d)$, which means no re-protection will be triggered. Therefore, the formula is valid.

Further more, $\widehat{dist}(x,d) = \widehat{dist}(s,d) - \widehat{dist}(s,x)$ and $dist(f,d) = dist(s,d) - dist(s,f)$, both of which can be directly obtained without any extra calculation. As for $dist(x,f)$, if $x = q$, $dist(x,f) = dist(s,q) - dist(s,f)$, because q is a descendant of f in the old SPT. However, if $x = p$ ($V_{DF} = 0$), s have to perform a SPT calculation rooted at p, which introduces nontrivial overhead for s. Another more rigorous condition can be used to decide whether $V_{repro} = 0$ when $V_{DF} \neq 0$:

$$\widehat{dist}(p,d) - dist(s,d) \leqslant dist(p,s) - dist(s,f)$$

Now we prove that the condition guarantees that f is not on the route from p to d by Lemma 2.

Lemma 2. *The condition* $\widehat{dist}(p,d) - dist(f,d) \leqslant dist(p,s) - dist(s,f)$ *ensures that* f *is not on* $route(p,d)$.

Proof. We prove the lemma by reduction to absurdity. Assume that f is on the route from p to d. Let n be the prior node of f on the route from p to d. Then

$$\widehat{dist}(p,d) \geqslant dist(p,d) = dist(p,n) + dist(n,f) + dist(f,d).$$

However, since f is not on $route(s,p)$, we have

$$dist(s,p) < dist(s,f) + dist(f,n) + dist(n,p).$$

Combine the above two inequalities using

$$dist(p,n) + dist(n,f) = dist(f,n) + dist(n,p),$$

we have $\widehat{dist}(p,d) - dist(f,d) > dist(s,p) - dist(s,f)$, which is contradictory with the original condition. Therefore, the original condition ensures that f is not on $route(p,d)$. □

So far, we have provided the calculation method for Ω. In the following subsections, we introduce the two stages of the algorithm MPCT.

4.2. First stage: MPCT construction

Table 2 summarizes the notations used in the algorithm. $\mathcal{T}$ is the given SPT rooted at s. All the operations in the algorithm are based on this SPT. ω is used to replace the above Ω.

Algorithm 1. First stage: find the incoming node. ($\mathcal{T}$ is the SPT rooted at s and f is the failed node.)

```
1: Set all nodes in $D$ as *floating*
2: **for all** $e \in I(D)$ **do**
     ▶ Initialize the Priority Queue
3:   **if** $S(e).state \neq floating$ **then**
4:     $dist \leftarrow S(e).dist + lenght(e)$
5:     Set $\omega$ according to our method
6:     enqueue$(Q, \{E(e), (S(e), dist, \omega)\})$
7:   **end if**
8: **end for**
9: **while** $Q \neq \emptyset$ **do**
10:    $\{d, (p, dist, \omega)\} \leftarrow$ extractMin $(Q)$
11:    Move $ATTree(d)$ under $p$
12:    **for all** $x \in ATTree(d)$ **do**
13:      Set $x.innode$ and $x.nhop$ accordingly
14:      $x.dist \leftarrow p.dist + \widehat{dist}(p,x)$
15:      $x.state \leftarrow attached$
16:      **if** $x \in Q$, remove $x$ from $Q$
17:  **end for**
18:  **for all** $e \in O(ATTree(d))$ **do**
19:      **if** $E(e).state = floating$ **then**
20:        $dist \leftarrow S(e).dist + lenght(e)$
21:        Set the $\omega$ according to our method
22:        enqueue$(Q, \{E(e), (S(e), dist, \omega)\})$
23:    **end if**
24:  **end for**
25: **end while**
```

Algorithm 1 shows the first stage of Tunnel-AT routing algorithm. Given the shortest path tree $\mathcal{T}$ rooted at s and a failed node f, this algorithm constructs the MPCT and finds incoming node for each f-affected node.

Two functions of the priority queue are used in the algorithm and need extra explanation:

- enqueue$(Q, \{x, (p, d, \omega)\})$, adds an item to the priority queue Q. If an item of node x exists in Q, the new item replaces the old one when the new attribute ω is smaller.
- extractMin(Q), extracts the item with the smallest ω from the priority queue Q.

As shown in Algorithm 1, lines from 1 to 8 set the states of f-affected nodes as *floating* and initialize the priority queue, where the affected nodes with unaffected neighbors are added in the priority queue. From line 9 to line 25, the ATTree with the minimized Ω is found and attached below a new parent in each iteration. As we have introduced the methods to calculate the value of Ω, details are ignored in the algorithm. For each node in the ATTree, the corresponding next hop (*hhop*), incoming node (*innode*), dist in the MPCT (*dist*) and state (*state*) are set accordingly, which will be used to generate the protection route in the second stage.

4.3. Second stage: protection route computation

Algorithm 2 shows the second stage of our algorithm, which computes the protection route for each f-affected destination based on the MPCT from Algorithm 1.

The protection route for a destination d is a tuple $< TEP, DF >$, where TEP is the attaching node (the tunnel end point) for d and DF is the incoming node (the root of the corresponding super AT-Tree). According to the above explanations, if $V_{path} < 0$, TEP will not reroute the packet back to s, thus DF is not needed and $DF = null$.

Given that V_{path} is calculated in Algotithm 1, the complexity of Algorithm 2 is $O(|D|)$, where D is the set of nodes affected by the failure of f.

Algorithm. 2 Compute the backup routes

```
1: for all $d \in D$ do
2:     $DF \leftarrow d.innode$
3:     $TEP \leftarrow DF.parent$
4:     if $V_{path} \geqslant 0$ then
   ▶ DF is necessary
5:         Set the protection route as $< TEP, DF >$
6:     else                          Tunnel without DF is enough
7:         State Set the protection route as $< TEP, null >$
8:     end if
9: end for
```

4.4. Complexity analysis: MPCT and other IPFRR schemes

We conclude the section by analyzing the complexity of MPCT. We also provide the analysis of other IPFRR schemes including LFA [7], Tunnel [10] and NotVia [9]. As the intra-domain routing protocol (i.e., OSPF) employs the algorithm of shortest path first (SPF) to compute the optimal next hop for each intra-domain destination, we use SPF as a metric to compare the complexities of these IPFRR schemes.

We first show that the complexity of MPCT by Theory 2.

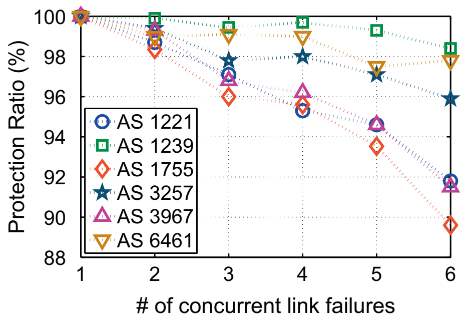

Fig. 9. Protection ratio for multi-link failures.

Theorem 2. *The time complexity of MPCT is* $O(|\mathbf{E}|\log|\mathbf{V}|)$ *and quantitatively less than the algorithm of SPF.*

Proof. Each intra-domain router has to perform the algorithm of shortest path first (SPF) to construct the optimal next hop for each intra-domain destination. Based on the Shortest Path Tree (SPT) computed by SPF, the router can construct protection paths for all destinations by k times (k is the number of neighbors) of the above algorithms (Algorithms 1 and 2) with one neighbor hypothetically failing each time. These algorithms are based on operations of priority queue (Q).

SPF invokes three types of priority queue operations: *add*, *extractMin*, and *decreaseKey*. MPCT algorithms invoke one extra type of operation – *remove*. Note that depending on whether an item is already in Q, *enqueue* would invoke *add* or *decreaseKey*. The complexity of SPF is $O(|\mathbf{E}|\log|\mathbf{V}|)$ [21]. Let V_i denote the number of affected nodes if neighbor i fails. Let E_i denote the number of edges attached to one of these nodes. The number of operations in MPCT for this neighbor i can be expressed as follows:

- Maximum number of *extractMin*: l, # of *ATTrees*.
- Maximum number of *add*: V_i.
- Maximum number of *remove*: $V_i - l$.
- Maximum number of *decreaseKey*: E_i.

Note that the dominating operation is *decreaseKey*. Therefore, the complexity of MPCT is

$$\sum_{i=1}^{k} E_i \log V_i \leqslant \sum_{i=1}^{k} E_i \log|\mathbf{V}| = \log|\mathbf{V}| \sum_{i=1}^{k} E_i = O(|\mathbf{E}|\log|\mathbf{V}|),$$

which is the same as one full SPF. Since $V_i < |\mathbf{V}|$, the average queue length for MPCT when performing queue operations is less than that of a full SPF. Thus, the complexity of MPCT algorithm is less than one full SPF. □

LFA, NotVia and Tunnel respectively need k, $|\mathbf{V}| - 1$ and $|\mathbf{V}| - 1$ times of SPFs to compute backup next hops for all the intra-domain destinations (see Section 6 for detailed analysis). The major obstacle of deploying Tunnel and NotVia is the higher complexity. They introduce new bottlenecks in route computing. In contrast, LFA has already been accepted as a standard in IETF because of its lower complexity. MPCT has even lower complexity and higher protection ratio than LFA, which would make it more attractive.

5. Simulation

5.1. Simulation setup

To demonstrate the performance of our mechanism and algorithm, we make comprehensive simulations on three different sets

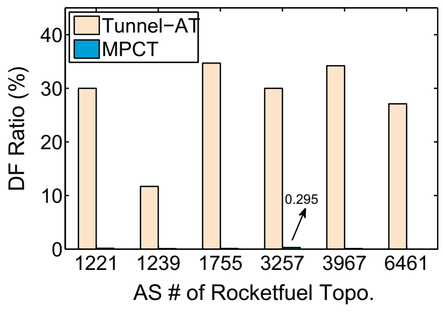

Fig. 10. DF ratios (Tunnel-AT and MPCT) in the Rocketfuel topologies.

of topologies, including BRITE-generated topologies and real topologies.

- We obtain the real topology from China Education and Research Network (CERNET) [15]. CERNET is a medium-scale AS with about 1500 institutions connected.
- We obtain six backbone intra-domain topologies with inferred link weights from Rocketfuel Project [16]. We extract the biggest biconnected components from these topologies, since non-biconnected components can not be protected and will add noise data to the simulation results. The topological characteristics of these six ASes are summarized in Table 3.
- For more comprehensive analysis, we generate diverse topologies by BRITE [17]. The number of nodes is set from 50 to 500. The parameter *m* (links/new node) is set from 2 to 14. Other parameters for the BRITE topology generation is in Table 4.

5.2. Simulation results

Before diving into the details of the simulation results, we first make some explanations of our implementations of LFA, NotVia and Tunnel. Although there might be multiple protection paths in LFA and Tunnel, to accelerate the computing process, we terminate the algorithms when any available protection path is found. Besides, a failure might separate the topology and cause some unreachable destinations. The path to an unreachable destination after the failure can not be protected by any scheme. We only focus on those paths that *can* be protected.

5.2.1. Computation complexity

According to the theoretical analysis in Section 4.4, MPCT has the complexity of less than one SPF, which is a great advantage compared with LFA, NotVia and Tunnel. To further confirm this advantage, we make comprehensive simulations with BRITE-generated topologies. We run the algorithms of MPCT, LFA, NotVia and Tunnel on an Intel Core Duo CPU of 2.00 GHz with RAM 2.0 GB. As Fig. 5 shows, MPCT requires the shortest computing time, which is around one time of SPF. LFA and Tunnel consume respectively about k (the degree of the topology) and $|\mathbf{V}|$ (the size of the topology) times of SPFs, which are both consistent with our theoretical analysis. For the larger topologies with 500 nodes, NotVia has the same property as Tunnel (Fig. 5(b)). However, the running times in the smaller topologies are less than $|\mathbf{V}|$ times of SPF (Fig. 5(a)). This is because, NotVia computes extra $|\mathbf{V}| - 1$ SPTs with one hypothetically failed node each time. A smaller topology is more likely to be separated by a failure and we do not compute protection paths for the unreachable nodes, which might lead to a significant decrease in computing time.

As Fig. 6 shows, the results on the real topologies of Rocketfuel are also consistent with our theoretical analysis. AS 1229 is the largest and densest topology, on which LFA, Tunnel and NotVia consumes the maximum times of SPF. As for MPCT, the computing times are less than one SPF on all the six topologies.

5.2.2. Protection coverage

We follow the definition of protection ratio in [22]. A node is protected iff all the intra-domain paths containing the node are protected. The protection ratio is the number of protected nodes divided by the number of nodes in the topology.

As Fig. 7(a) and (b) show, the protection ratios of MPCT and NotVia are both 100% in all diverse topologies. The protection ratios of Tunnel in the topologies with degree of four and eight are respectively around 96% and 100%; the protection ratios of LFA are respectively 28% to 45% and 50% to 90%. Note that Tunnel and LFA perform better in denser topologies, which is because there are more available paths in denser topologies. Besides, the performance of LFA degrades as the topology size increases. The reason is that, in larger topologies, a node failure involves more paths, any one of which might be unprotected.

In Fig. 8, we provide the results of protection ratio on Rocketfuel topologies. We can see that the protection ratios of MPCT and NotVia are still 100%, while the results of LFA and Tunnel are also consistent with BRITE.

In order to demonstrate the performance of MPCT further, we evaluate the protection coverage in the case of multiple concurrent failures. As multi-node failures would probably leads to topology separation, we use multi-link failures instead. In the simulation, to avoid loop, a protected packet before decapsulation will be dropped if encountering another failure. As shown in Fig. 9, MPCT averagely achieves about 94% protection ratio even when there are six randomly failed links in the topology.

5.2.3. Direct forwarding ratio

As direct forwarding (DF) is not well supported by ISPs, it should be avoided as far as possible to guarantee a better performance of MPCT without DF. Direct Forwarding ratio is the ratio of the protection paths using DF to all the protection paths. As shown in Fig. 10, the DF ratios of MPCT are no more than 1% (compared with 30% of Tunnel-AT) in all the six Rocketfuel topologies. This demonstrates that even without direct forwarding, MPCT can provide more than 99% protection coverage for single-node failures.

For more comprehensive analysis, we use the BRITE topologies. As shown in Fig. 11, more than 99% protection paths of MPCT are DF-free. For the dense topologies with the degree of ten, the DF ratios are zero, which means MPCT can provide 100% protection

	Size: 100	Size: 150	Size: 200	Size: 250	Size: 300	Size: 350
Avg. Degree: 4	0.0737%	0.2851%	0.1643%	0.1192%	0.2328%	0.1023%
Avg. Degree: 6	0.0108%	0.0000%	0.0078%	0.0082%	0.0137%	0.0067%
Avg. Degree: 8	0.0110%	0.0047%	0.0000%	0.0017%	0.0023%	0.0000%
Avg. Degree: 10	0.0000%	0.0000%	0.0000%	0.0000%	0.0000%	0.0000%

Fig. 11. The DF Ratios of Brite Topologies with average degrees from 4 to 10 and sizes from 50 to 350.

	Size: 100	Size: 150	Size: 200	Size: 250	Size: 300	Size: 350
Avg. Degree: 4	0.1684%	0.1425%	0.1592%	0.0898%	0.1006%	0.1322%
Avg. Degree: 6	0.0108%	0.0000%	0.0026%	0.0082%	0.0114%	0.0025%
Avg. Degree: 8	0.0000%	0.0000%	0.0000%	0.0000%	0.0000%	0.0008%
Avg. Degree: 10	0.0000%	0.0000%	0.0000%	0.0000%	0.0000%	0.0000%

Fig. 12. The Re-Protection Ratios of Brite Topologies: average degree from 4 to 10; size from 50 to 350.

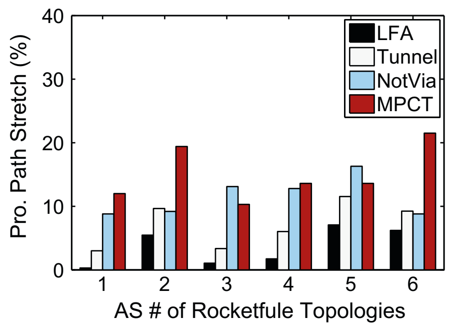

Fig. 13. Protection path stretch in the Rocketfuel topologies.

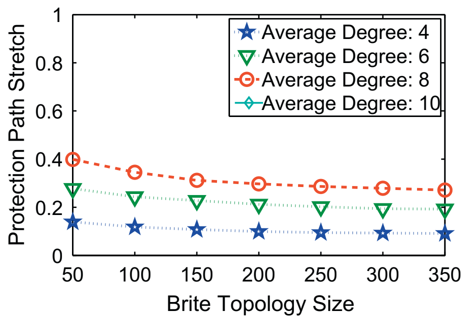

Fig. 14. Protection path stretch in the Brite topologies.

coverage without direct forwarding. This is because MPCT prefers the DF-free protection path and denser topology provides more DF-free protection paths.

In summary, if direct forwarding is supported by the ISP, MPCT can provide 100% protection coverage for single-node failures; if not, MPCT can still achieve more than 99% protection coverage. This guarantees that MPCT can work effectively in the practical network environment where direct forwarding is probably not supported.

5.2.4. Re-protection ratio

We now study the re-protection ratio of MPCT. Re-protection ratio is the ratio of the protection paths with re-protection to all the protection paths. Although re-protection is always supported by routers with tunnel function, it introduces some extra overhead. Therefore, one design goal of MPCT is avoiding re-protection as far as possible. The simulation results show the overhead caused by re-protection can be nearly ignored.

Fig. 12 shows the re-protection results in diverse BRITE topologies. The re-protection ratios are all below 1% without exception. For the denser topologies, the re-protection ratios are smaller. This can be explained similarly to the same phenomenon of DF ratio: the denser topology has more selectable protection paths, thus the protection path without re-protection are more possible to exist. We obtain analogous results on Rocketfuel topologies.

5.2.5. Protection path stretch

We next study the protection path stretch, which is inevitable in all IPFRR schemes. As shown in Fig. 13, LFA performs better than other schemes. The protection path stretches of MPCT are are in the range between 10% and 20%, which is at the same level as NotVia, but twice of Tunnel. However, we believe that this sacrifice is acceptable compared with the achievement of MPCT in reducing the computing complexity.

As for the Brite topologies, some interesting results can be observed in Fig. 14. As the topology size grows, the protection path stretch degrades. We believe that this is because routing protection is a local behavior and the protection path stretch for a remote destination is smaller than a nearby destination. As the average degree degrades, the stretch also degrades. This can be explained as follows: the smaller the average degree is, the more possible the protection path is the optimal (or closer to the optimal) because there are fewer available protection paths.

5.2.6. CERNET result

For the CERNET topology, the results are consistent with Rocketfuel and BRITE topologies. The protection ratios of LFA, Tunnel, NotVia and MPCT are respectively 24%, 35%, 100% and 100%; the protection path stretches are respectively 0.62%, 3.5%, 11% and 20%; the computation time of MPCT is still the shortest among the four schemes.

6. Cons and pros of Fast Reroute Schemes

There are two main approaches for network survivability, namely protection and restoration [4–6]. *Protection* is based on pre-computed backup paths: a protection (backup) path is pre-computed to forward the traffic if the primary path fails. *Restoration* computes a new path after a failure has occurred. We focus on protection, which has proved to be an efficient approach for improving the intra-domain network performance. Among these protection schemes, two main directions exist: IP fast reroute (IPFRR) and multi-path routing (MPR).

In this section, we provide a comparison of these schemes and MPCT. As summarized in Table 5, LFA supports router-level incremental deployment because it does not involve tunnel or extra configuration, while the other schemes must be deployed in the whole AS. Besides, LFA has a low computing complexity. However, LFA cannot provide a satisfactory protection coverage. Tunnel and NotVia are not practical because of their high computing complexity; while multiple routing configurations (MRC) [23] and NotVia involve nontrivial memory overhead. With a minor increase in protection path stretch, MPCT achieves a full (or almost full without direct forwarding) protection coverage, a low memory overhead and the lowest computing complexity. In what follows, we provide the detailed analysis of these schemes.

6.1. IP Fast Reroute

IP fast reroute (IPFRR) schemes [7–10] are well compatible with the currently used intra-domain routing protocol OSPF, and they have no impact on the normal routing: the protection path is only

Table 5
Comparison of Fast Reroute Schemes.

Schemes	Pro. coverage	Memory	Computing	Stretch	Deployment
LFA	Low	Low	Low	3%	Router-level
Tunnel	High	Low	High	7%	AS-level
NotVia	Full	Heavy	High	12%	AS-level
MRC	Full	Medium	Medium	–[a]	AS-level
MPCT	(≈) Full [b]	Low	Lowest	15%	AS-level

[a] We did not simulate this scheme.
[b] Full with direct forwarding or almost full without direct forwarding.

used when the normal optimal path fails. Therefore, they attract great attention from the academic field.

Among all the IPFRR schemes, Loop Free Alternate (LFA) [7] is the simplest one. In LFA, the protection source node finds an available neighbor as an alternate instead of the failed next hop. To find a potential backup next hop from the neighbors, a router needs to compute the SPTs rooted at the neighbors, Therefore, the complexity of LFA is k times of SPF. As LFA does not involve tunnel or extra configuration, it supports router-level incremental deployment, i.e., a router benefits all from its own deployment without other routers' cooperation. For its low complexity and router-level incremental deployability, LFA has become an Internet RFC. However, LFA provides only about 40% protection coverage for single-node failures [13], which is far from the requirements of Internet Service Providers (ISPs). A novel method of changing the topology (without changing the general shortest paths) to achieve 100% protection coverage with LFA is introduced in [24], whose feasibility is under confirmation. Therefore, schemes with higher protection coverage are in request.

Efficient Scan for Alternate Paths (ESCAP) [8], another IPFRR scheme, can provide 100% protection coverage for single-link/node failures. However, the complexity for backup route calculation of ESCAP is $O(|V|^3)$. As the complexity of shortest path first (SPF) algorithm is $O(|V|^2)$ (the simplest implementation [21]), the complexity of ESCAP is $|V|$ times of SPF. (The complexity of MPCT is less than one SPF).

IP fast reroute using NotVia addresses (NotVia) [9] is a novel scheme that also provides 100% protection coverage for single-link/node failures. In NotVia, the protection source node reroutes the packet to the next next hop (the downstream node of the failed node) by the special NotVia address. The router has to compute routes for $2|\mathbf{E}|$ extra NotVia destinations, which requires $|\mathbf{V}| - 1$ times of SPFs with a router hypothetically failing each time. Besides, the memory overhead caused by NotVia addresses is nontrivial [25,26,20]. These two severe problems block NotVia from practical deployment. In [25], NotVia aggregation is used to alleviate the memory overhead. However, the assumption that each router is configured with an IP prefix covering both its interface addresses and NotVia addresses is unrealistic. Besides, their improvement only alleviates the complexity of NotVia algorithm but does not change the order of the complexity. Lightweight NotVia [26], another significant optimization over NotVia, decreases the memory overhead of NotVia dramatically by redundant trees [27]. However,in this mechanism, the protection source node needs to forward two copies of one packet to the next next hop by two different routes. This approach would aggravate network congestion during failure by wasting bandwidth.

Tunnel [10], which has been discussed in Section 2.1, provides higher protection coverage and involves not memory overhead, However, to find an available tunnel end point for each destination, the router needs to construct a reverse SPT rooted at the destination, which is equal with SPF in complexity. Therefore, $|\mathbf{V}| - 1$ SPFs are required, which blocks it from practical deployment.

In multiple routing configurations (MRC) [23], each router maintains multiple backup disjoint routing tables (configurations). For each failure, at least one configuration can bypass the failure. MRC involves nontrivial memory overhead as a routing table is required for each backup configuration. Besides, a mark for the selected configuration is inserted into packet headers to notify the other routers to keep consistent. MRC provides 100% protection coverage. However, the overhead of maintaining multiple routing tables and extending packet headers blocks it from practical deployment.

Recently, Atlas, et al., proposed a new IPFRR scheme of Maximally Redundant Tree (MRT) ([11,12]). MRT keeps the shortest path as the primary route and uses two MRT routes (blue and red) as backup routes. Similar to NotVia, MRT also uses tunnel to send the failure-affected packets to a safe node downstream the optimal path. Each MRT root introduces two extra addresses (blue or red) into the intra-domain network, which causes extra burden for intra-domain routers.

6.2. Multi-path routing

In multi-path routing (MPR) [28–30], multiple paths to the same destination are also pre-computed before the failure. These pre-computed paths do not have the master–slave relationship like IPFRR. All paths might be used before failure occurs. Generally, none of them is optimal in length. Besides, some of the MPR schemes are even centralized, and thus are not compatible with the current intra-domain routing protocol OSPF.

7. Conclusion

In this paper, we present the IPFRR scheme of Minimum Protection Cost Tree (MPCT), which is based on IP-in-IP tunnel as NotVia, MRT and Tunnel. MPCT provides 100% (or almost 100% without direct forwarding) protection coverage for single-link/node failures. The complexity of MPCT is below one full SPF calculation. Compared with the other schemes, including LFA, Tunnel and NotVia, MPCT reduces the computational complexity dramatically. Even without direct forwarding, MPCT provides more than 99% protection coverage; besides, MPCT has lower complexity than LFA, which provides only about 40% protection coverage. As LFA has already been accepted by IETF as a standard, we believe that MPCT provides another choice for ISPs to enhance the network performance during failures.

Acknowledgement

This work was supported by the National Natural Science Foundation of China under Grant No. 61073166 and 61133015; by the National Basic Research Program of China (973 Program) under Grant No. 2009CB320502; and by the National High-Tech Research and Development Program of China (863 Program) under Grant No. 2011AA01A101.

References

[1] C. Labovitz, A. Ahuja, A. Bose, F. Jahanian, Delayed internet routing convergence, in: Proc 2000 ACM SIGCOMM, Stockholm, Sweden.
[2] P. Francois, C. Filsfils, J. Evans, O. Bonaventure, Achieving sub-second IGP convergence in large IP networks, ACM SIGCOMM CCR 35 (3) (2005) 35–44.
[3] K. Lakshminarayanan, M. Caesar, M. Rangan, T. Anderson, S. Shenker, I. Stoica, Achieving convergence-free routing using failure-carrying packets, in: Proc. 2007 ACM SIGCOMM, Kyoto, Japan.
[4] A. Fumagalli, L. Valcarenghi, IP restoration vs. WDM protection: is there an optimal choice?, IEEE Network 14 (6) (2000) 34–41
[5] L. Sahasrabuddhe, S. Ramamurthy, B. Mukherjee, Fault management in IP-over-WDM networks: WDM protection versus IP restoration, IEEE JSAC 20 (1) (2002) 21–33.

[6] G. Iannaccone, C. Chuah, S. Bhattacharyya, C. Diot, Feasibility of IP restoration in a Tier-1 backbone, IEEE Network 18 (2) (2004) 13–19.
[7] A. Alia, Z. Alex, T. Raveendra, C. Gagan, M. Christian, I. Brent, F. Don, Basic Specification for IP Fast-Reroute: Loop-free Alternates, IETF RFC 5286 (Sep. 2008).
[8] K. Xi, H.J. Chao, IP fast rerouting for single-link/node failure recovery, in: Proc 2007 IEEE Broadnets, Raleigh, USA.
[9] M. Shand, S. Bryant, S. Previdi, IP fast reroute using not-via addresses, IETF Draft, draft-ietf-rtgwg-ipfrr-notvia-addresses-05, Mar. 2010.
[10] S. Bryant, C. Filsfils, S. Previdi, M. Shands, IP Fast Reroute using tunnels, IETF Draft, draft-bryant-ipfrr-tunnels-03 (Sep. 2007).
[11] A. Atlas, R. Kebler, M. Konstantynowicz, G. Enyedi, A. Csaszar, R. White, M. Shand, An architecture for IP/LDP Fast-Reroute using maximally redundant trees, IETF Draft, draft-atlas-rtgwg-mrt-frr-architecture-01 (Oct. 2011).
[12] A. Atlas, G. Enyedi, A. Csaszar, Algorithms for computing maximally redundant trees for IP/LDP Fast- Reroute, IETF Draft, draft-enyedi-rtgwg-mrt-frr-algorithm-00 (Oct. 2011).
[13] P. Francois, O. Bonaventure, An evaluation of IP-based fast reroute techniques, in: Proc 2005 ACM CoNEXT, Toulouse, France.
[14] P. Narváez, K.-Y. Siu, H.-Y. Tzeng, New dynamic SPT algorithm based on a ball-and-string model, IEEE/ACM TON 9 (6) (2001) 706–718.
[15] The China Education and Research Network (CERNET), <http://www.edu.cn/english>
[16] R. Mahajan, N. Spring, D. Wetherall, T. Anderson, Inferring link weights using end-to-end measurements, in: Proc 2002 IEEE IMW, Marseille, France.
[17] BRITE, <http://www.cs.bu.edu/brite/>
[18] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, C. Diot, Characterization of failures in an IP backbone, in: Proc. IEEE INFOCOM, Hong Kong, 2004.
[19] L. Pan, M. Xu, Q. Li, D. Jen, Lightweight IP Fast Reroute with Tunnel-AT, in: Proc. IEEE/ACM IWQoS (Poster), Beijing, PR China, 2011.
[20] Q. Li, M. Xu, Q. Li, D. Wang, Y. Cui, IP Fast Reroute: notvia with early decapsulation, in: Proc. IEEE GlobeCom Miami, USA, 2010.
[21] D. Algorithm, <http://en.wikipedia.org/wiki/DijkstraAlgorithm>
[22] P. Francois, O. Bonaventure, An evaluation of IP-based Fast Reroute techniques, in: Proc. ACM CoNext, Toulouse, France, 2005.
[23] A. Kvalbein, A.F. Hansen, T. Ččic, S. Gjessing, O. Lysne, Multiple routing configurations for fast IP network recovery, IEEE/ACM Trans. Network 17 (2009) 473–486.
[24] G. Rvri, J. Tapolcai, G. Enyedi, A. Csszr, IP Fast Reroute: loop free alternates revisited, in: Proc. IEEE Infocom, Shanghai, PR China, 2011.
[25] A. Li, P. Francois, X. Yang, On improving the efficiency and manageability of notvia, in: Proc. CoNEXT ACM, New York, USA, 2007.
[26] G. Enyedi, P. Szilgyi, G. Rtvri, A. Csszr, IP Fast ReRoute: lightweight not-via without additional addresses, in: Proc. IEEE INFOCOM, Rio de Janeiro, Brazil, 2009.
[27] M. Médard, S.G. Finn, R.A. Barry, Redundant trees for preplanned recovery in arbitrary vertex-redundant or edge-redundant graphs, IEEE/ACM Trans. Network 7 (5) (1999) 641–652, http://dx.doi.org/10.1109/90.803380.c.
[28] Y.O. Lee, A.L.N. Reddy, Disjoint multi-path routing and failure recovery, in: Proc. ICC, Cape Town, South Africa, 2010.
[29] A. Greenberg, G. Hjalmtysson, D.A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, H. Zhang, A clean slate 4D approach to network control and management, SIGCOMM CCR 35 (2005) 41–54.
[30] H. Peterson, S. Sen, J. Chandrashekar, L. Gao, R. Guerin, Z. Zhang, Message-efficient dissemination for loop-free centralized routing, ACM SIGCOMM CCR 38 (2008) 63–74.