Abstract—In today’s IP network, if a link or router fails, packets that traverse the failed link or router will be lost until the network re-converges even if there exists another path bypassing the failure. We call such a single failure a valid single failure. Therefore, the IETF published a framework, called IP Fast ReRoute (IPFRR), which aims to provide protection against such valid single failures before the network re-converges. Based on the IPFRR framework, a lot of methods have been proposed. One category of IPFRR methods is IP-tunnel-based. The IP-tunnel-based IPFRR methods impose extra cost, such as the encapsulation cost, on the traffic delivery. In this paper we focus on the other category of IPFRR methods, no-tunnel-based IPFRR methods, which do not impose any extra cost on the traffic delivery. However, the existing no-tunnel-based IPFRR methods cannot provide complete protection against all the valid single failures. Therefore, in this paper, we propose a new kind of no-tunnel-based IPFRR method, named with RPFP, which can provide complete protection against all the valid single failures.

Keywords—IP fast reroute; failure recovery; resilient routing

I. INTRODUCTION

The Internet has been changing from transferring text data to multimedia data over the past years. The multimedia applications, such as VoIP (Voice over IP), are sensitive to packet losses. However, in today’s IP network, if a link or router fails, packets that traverse the failed link or router will be lost until the network re-converges even if there exists another path bypassing the failure. We call such a single failure a valid single failure. Specifically, the valid single failure case is defined as follows. Given a single failure case (S, D, F), where S is the source, D is the destination and F is the single failure, we say this single failure case is a valid single failure case if and only if the traffic delivery from S to D is disrupted by the failure of F and S is still connected to D under the failure of F.

Therefore, the IETF published a framework [1], called IP Fast ReRoute (IPFRR), which aims to provide protection against such valid single failures before the network re-converges. Based on the IPFRR framework, a lot of IPFRR methods have been proposed. One category of IPFRR proposals is IP-tunnel-based, such as [2,3,4]. The key idea of the IP-tunnel-based proposals is to make the affected traffic bypass the single failure by using IP tunnels. The IP-tunnel-based IPFRR methods can provide complete protection against all the valid single failures under the universal deployment, but if using tunnels, packets will be encapsulated with extra IP headers, consuming extra bandwidth. In addition, if packets are encapsulated with extra IP headers, the packets will be fragmented so as not to exceed the MTU (Maximum Transmission Unit), increasing the burden on routers. Moreover, it is complex to manage and configure tunnels. Thus, in this paper we focus on the other category of IPFRR methods, no-tunnel-based IPFRR methods, which do not impose any extra cost on the traffic delivery. However, the existing no-tunnel-based method methods, which are LFA [5] and U-turn [6], cannot provide complete protection against all the valid single failures even under the universal deployment. Using simulations, LFA and U-turn can separately protect on average 69% and 91% of all the valid single failure cases under the universal deployment.

Therefore, in this paper, we propose a new kind of no-tunnel-based IPFRR method, named with Reverse Path Forwarding Protection (RPFP), which can provide complete protection against all the valid single failures under the universal deployment. RPFP only needs to make two necessary modifications, modifying the forwarding logic of the router and adding an algorithm of computing the backup routing, which must be made by any IPFRR methods including the IP-tunnel-based IPFRR methods. The key insight of RPFP is that the traffic affected by a failure is reversely forwarded until joining the normal routing. A distinct characteristic of packets going along the reverse forwarding path is: those packets come to routers from their primary next hops. Therefore, a packet going along the reverse forwarding path can be easily identified by checking whether the packet comes from the port that it should have been forwarded to. The contributions of this paper are as follows.

(a) We design two RPFP methods, RPFP-1 and RPFP-2. RPFP-1 can provide complete protection against only all the valid single link failures; RPFP-2 can provide complete protection against both all the valid single link failures and all the valid single node failures, but RPFP-2 needs higher computational complexity than RPFP-1. The computational complexity of RPFP-1 is \(O(V^2E\log(E))\) while the computational complexity of RPFP-2 is \(O(V^2V^*E)\), where \(V\) is the number of nodes and \(E\) is the number of links in the network. The network operators can choose to use RPFP-1 or RPFP-2 according to their own situations.

(b) We not only prove their respective complete protection performance for RPFP-1 and RPFP-2 in theory, but also comprehensively compare RPFP-1 and RPFP-2 with the other existing no-tunnel-based IPFRR methods.

(c) We evaluate the performance of the no-tunnel-based IPFRR methods under partial deployment for the first time in...
our view. From simulations, we find that if only 40% of routers are upgraded to deploy RPFP-1 or RPFP-2, more than 80% of valid single failure cases will be protected. Both RPFP-1 and RPFP-2 can protect much more single failure cases than the other existing no-tunnel-based methods under any deployment ratio.

The rest of this paper is organized as follows. In Section II, we will introduce the related work. In Section III, we will describe the tasks and limitations of this paper. In Section IV, we will describe RPFP-1 and RPFP-2. In Section V, we will evaluate RPFP-1 and RPFP-2 and compare them with the other existing no-tunnel-based IPFRR methods. In Section VI, we will address the partial deployment issue. In Section VII, we will conclude this paper and present our future work.

II. RELATED WORK

In this paper, we focus on no-tunnel-based IPFRR methods. However, the existing no-tunnel-based IPFRR methods cannot provide complete protection against all the valid single failures. Therefore, in this paper, we propose a new kind of no-tunnel-based IPFRR method, named with RPFP, which can provide complete protection against all the valid single failures. LFA [5] and U-turn [6] are the two existing no-tunnel-based IPFRR methods. We will comprehensively compare PRFP with LFA and U-turn, so we describe them in detail.

The key idea of LFA is to reroute the affected traffic through the pre-computed loop-free alternative next hop when a failure is detected. The key component of LFA is to compute the loop-free alternative next hops. A loop-free alternative next hop is to assure no routing loops during convergence. The criterion of a neighbor N being a loop-free alternative next hop of S towards D is as follows:

\[
\text{cost}(N, D) < \text{link}(N, S) + \text{cost}(S, D)
\]

where cost(i, j) denotes the cost from i to j; link(p, q) denotes the link weight from p to q.

U-turn is another no-tunnel-based IPFRR method. The key idea of U-turn is to preferentially use the loop-free alternative next hop to reroute the affected traffic when a failure is detected. If a router S has no loop-free alternative next hop towards a destination D, but a neighbor N of S has loop-free alternative next hops towards D, S will forward the affected traffic to N. Obviously, U-turn can protect more valid single failure cases than LFA, but U-turn still cannot provide complete protection against all the valid single failures.

Besides the above no-tunnel-based IPFRR methods, there are some other literatures relative to failure recovery as follows. In [7], the no-tunnel-based IPFRR methods, LFR and U-turn, were evaluated. Because the implementation of tunnels is complex, the IP-tunnel-based IPFRR methods were simplified for evaluations in [7]. Obviously, the simplified IP-tunnel-based IPFRR cannot provide complete protection against all the valid single failures. In [8], the convergence time of the link-state-protocol-based network is reduced to order of hundreds of milliseconds by carefully configuring parameters, which still cannot satisfy the requirements of multimedia applications for packet losses. In [9], the issue on how to add links in order to make LFA achieve complete protection was addressed. [10] is a survey of IP and MPLS fast reroute schemes. In [11,12], approaches to reducing the cost of Not-Via [3] were proposed. In [13,14], some micro-looping prevention techniques for IPFRR were proposed. In addition, there are many failure recovery methods proposed for the inter-domain routing, such as [15,16,17]. Moreover, there are many failure recovery methods proposed for other types of networks. For example, [18] was proposed for the wavelength division multiplexing (WND) network and [19] was proposed for the MPLS network.

III. TASKS AND LIMITATIONS

In this section, we will describe the tasks and limitations of this paper. Our work is based on the IPFRR framework. Our work retains the key idea of the IPFRR framework as follows. The pre-installed backup next hop is rapidly invoked once the failure of the primary next hop is locally detected, reducing the disruption time to the failure detection time on the local, which is generally a few tens of milliseconds. Moreover, our work is no-tunnel-based. The existing no-tunnel-based IPFRR methods, which are LFA and U-turn, cannot provide complete protection against all the valid single failures. Therefore, in this paper, we propose a new kind of no-tunnel-based IPFRR method, named with RPFP, in order to achieve complete protection against all the valid single failures.

Besides the contribution of designing the RPFP methods, we also comprehensively compare RPFP with the other no-tunnel-based IPFRR methods in this paper. Although the protection performance of LFA and U-turn was evaluated in [7], our evaluations are more comprehensive and fairer than [7]. In [7] single failure cases where sources are disconnected with destinations under failures are counted while we only count the valid single failure cases since those invalid single failure cases cannot be protected by any IPFRR methods. Thus, our evaluation case is fairer than the one in [7]. In addition, the protection performance only under the universal deployment was evaluated in [7]. We have not found any paper where the protection performance of the no-tunnel-based IPFRR methods under partial deployment was evaluated, so we evaluate that in this paper.

Now we introduce the limitations of this paper. Initially, for simplicity, the scope of the IPFRR framework has the following two limitations: (a) only the single failure protection is considered; (b) only the link-state-protocol-based network is considered. Our work retains the first limitation of the IPFRR framework because [20] shows that 70% of network failures at a time are single link failures. We will consider the protection against multiple unrelated failures in the future work. The second limitation of the IPFRR framework can be eliminated by using the centralized way as follows. A central controller collects the ubiquitous topology and routing information from each router by using SNMP (Simple Network Management Protocol), which is supported by almost all the routers. The central controller then computes the backup routing table for each router. The central controller finally downloads the computed backup routing tables to the corresponding routers. Besides the advantage of being able to work in non-link-state-
protocol-based intra-domain environment, the centralized way has another advantage, which is to reduce the computation cost of routers. Of course, RPFP can also work in the link-state-protocol-based network with the decentralized way because the link-state-protocol-based routers have the ubiquitous topology information. In this paper, we not only describe how RPFP works in the centralized way, but also describe how RPFP works in the decentralized way. Similarly, although LFA and U-turn were initially proposed for the link-state-protocol-based network in the centralized way, they can also work in the centralized way. Besides the above limitations, this paper has another limitation, which is that we only consider the situation that there is only one primary next hop for one destination. Expanding RPFP to support the situation that there are multiple primary equal-cost next hops (ECMP) for one destination is a big issue, which cannot be included in this paper. We will address that in the future.

IV. METHODOLOGY

In this section, we will introduce our RPFP methods. A RPFP method includes two key components. One is the forwarding logic and the other is the algorithm of computing backup next hops. We design two RPFP methods, RPFP-1 and RPFP-2. The only difference between RPFP-1 and RPFP-2 is on the algorithms of computing backup next hops, which we separately call the RPFP-1 algorithm and the RPFP-2 algorithm. RPFP-1 can provide complete protection against only all the valid single link failures. RPFP-2 can provide complete protection against all the valid single link and node failures, but RPFP-2 needs higher computational complexity than RPFP-1. In Subsection A, we will introduce some basic definitions that are often used in this paper. In Subsection B, we will describe the forwarding logic of RPFP. In Subsection C, we will describe the RPFP-1 algorithm. In Subsection D, we will describe the RPFP-2 algorithm. We describe the RPFP-1 and RPFP-2 algorithms in the centralized way as the main body, based on which, we describe how the RPFP-1 and RPFP-2 algorithms work in the link-state-protocol-based network with the decentralized way.

A. Basic definitions and theorems

A network topology can be described as an directed graph model $G=<V,E>$, where $V$ is the set of nodes (routers) and $E$ is the set of edges (links). In this paper, we think that node is equivalent to router, and edge is equivalent to link. $|V|$ is the number nodes. $|E|$ is the number of edges. Each edge is bi-directional. We use $Primary_{y}(S)$ to denote the primary next hop of Router $S$ towards $D$. Similarly, we use $Backup_{y}(S)$ to denote the backup next hop of Router $S$ towards $D$. We think the next hop is a router for simplicity in the context of this paper. We use a two-tuple $N1-N2$ to denote a link, where $N1$ and $N2$ are the two routers associated with the link. Now we define some terms that will be often used as follows. In Fig. 1, we give some examples for the following definitions.

**Definition 1:** Primary$_y^0(A)$ is the n$^{th}$ hop of $A$ towards $D$ using the primary routing. It is defined recursively as follows.

\[
\begin{align*}
\text{Primary}_y^0(A) &= \text{Primary}_D(\text{Primary}_y^{n-1}(A)) \\
\text{Primary}_y^0(A) &= A \\
\end{align*}
\]

\(n \geq 1\) \hspace{1cm} \(n = 0\)

**Definition 2:** The routing tree of a destination $D$. Given a network graph $G$, the set of edges $\{S - Primary_y(S) | S \in V$ and $S \neq D\}$ forms a sub-graph of $G$, denoted by $T$. If $T$ is a connected graph, $T$ must be a tree. It is because $T$ is a connected graph and the number of edges of $T$ is $|V|-1$. We call $T$ the routing tree of the destination $D$.

**Definition 3:** Father and Ancestor on the routing tree of a destination $D$. On the routing tree of $D$, Primary$_y(S)$ is called the father of $S$, and Primary$_y^n(S) \geq 1$ is called an ancestor of $S$. Reversely, $S$ is called a son of Primary$_y(S)$, and $S$ is called a descendant of Primary$_y^n(S) \geq 1$.

**Definition 4:** Tree edge and cross edge for a destination $D$. Given a network graph $G$, we call the edges that belong to the routing tree of $D$ to be tree edges and the other edges in $G$ to be cross edges.

**Definition 5:** Least Common Ancestor (LCA) of two nodes on the routing tree of a destination $D$. We suppose that the two nodes are $A$ and $B$, the LCA of $A$ and $B$ is the common ancestor that is the nearest to $A$ and $B$ on the routing tree of $D$. Specially, we think one node is an ancestor of itself here.

**Definition 6:** The triangle circle constructed by a cross edge for a destination $D$. We suppose that the cross edge is $A-B$ and the LCA of $A$ and $B$ on the routing tree of $D$ is $C$, then the path from $A$ to $C$ on the routing tree of $D$, the path from $C$ to $B$ on the routing tree of $D$ and the cross edge $B-A$ construct a circle, which we call the triangle circle constructed by Cross edge $A-B$ for $D$.

**Definition 7:** High-low relation with respect to a destination $D$. We say $A$ is higher than $B$ or $B$ is lower than $A$ with respect to $D$ if and only if $A=Primary_y^n(B) \geq 1$.

**Definition 8:** The peak of the triangle circle constructed by a cross edge for a destination $D$. We call the LCA of the two nodes associated with the cross edge to be the peak of the triangle circle.
Definition 9: The link protection ratio is the ratio of the number of valid single link failure cases that are protected to the number of the total valid single link failure cases.

Definition 10: The node protection ratio is the ratio of the number of valid single node failure cases that are protected to the number of the total valid single node failure cases.

B. The forwarding logic of RPFP

As shown in Fig. 2, each destination prefix has one primary next hop and one backup next hop in the forwarding table. The primary next hop towards each destination is generated by the normal routing protocols. The backup next hop towards each destination is generated by the RPFP algorithm. As shown in Fig. 2, when a packet comes, the longest matching prefix of the destination IP address of the packet is normally looked up in the forwarding table, if the primary next hop of the packet fails or the packet is from its primary next hop, which implies that a failure occurs on the path via its primary next hop towards the destination, so the packet will be forwarded via its backup next hop (if it exists); otherwise, the packet will be forwarded via its primary next hop. Obviously, RPFP does not impose extra look-ups.

C. The RPFP-1 algorithm

RPFP-1 can provide complete protection against all the valid single link failures. In this subsection, we will describe the RPFP-1 algorithm.

Given the triangle circle constructed by a cross link for a destination, we assign the backup next hop towards the destination for each router on the triangle circle in order to achieve the following goal. If any link on the triangle circle fails, according to the forwarding logic of RPFP, the affected traffic can be reversely forwarded along the triangle circle until turning across the cross link to the other side of the triangle circle, then joining the normal forwarding. We call such a path the protection path against the link failure. We summarize the assigning rule as follows.

Rule 1 (Assigning Rule). Given the triangle circle constructed by a cross link for a destination, we assign the backup next hop towards the destination for each router on the triangle circle as follows. Each of the two routers associated with the cross link should use the other router as its backup next hop towards the destination. The backup next hop of the peak router of the triangle circle should not be assigned in this turn. Each of the other routers on the triangle circle should use the far-from-peak router of its two adjacent routers on the triangle circle as its backup next hop towards the destination.

As shown in Fig. 3, the triangle circle constructed by Cross link D-G for Destination A is F-D-B-A-C-E-G-F. According to the assigning rule, the backup next hops of routers on the triangle circle towards A are assigned as indicated by the dash arrows. When Link D-B fails, (1) D can detect its primary next hop towards A fails, according to the forwarding logic of RPFP, D will forward the affected traffic to its backup next hop F; (2) F will receive the affected traffic from D, which is its primary next hop towards A, so according to the forwarding logic of RPFP, F will also invoke its backup next hop towards A, which is G, forwarding the affected traffic to G; (3) G will forward the affected traffic using the normal routing. Thus, the affected traffic is first forwarded along the reverse forwarding path D-F, and then turns across the cross link F-G, finally joins the normal routing towards A at G.

Given a destination, if a router is contained in only one triangle circle for the destination, the backup next hop of the router towards the destination can be assigned using the assigning rule according to the triangle circle the router is contained in. However, how about when a router is contained in two or more triangle circles for a destination? As shown in Fig. 4, the destination is A. Both Router D and Router B are contained in two triangle circles for A, which are the triangle circle D-B-C-D constructed by Cross link D-C for A and the triangle circle F-D-B-A-E-G-F constructed by Cross link F-G for A. The backup next hop of D towards A according to the triangle circle constructed by D-C collides with the one assigned according to the triangle circle constructed by F-
G. We separately give the two kinds of assignment in Fig. 4 (a) and Fig. 4 (b) as follows. In Fig. 4 (a), the backup next hop of D towards A is assigned according to the triangle circle constructed by D-C, whose peak is lower than the one of the triangle circle constructed by F-G. If so, the single failure of Link B-A cannot be protected as shown in Fig. 4 (a). However, in Fig. 4 (b), the backup next hop of D towards A is assigned according to the triangle circle constructed by F-G, whose peak is higher than the one of the triangle circle constructed by D-C. If so, the single failure of Link B-A can be protected as shown in Fig. 4 (b). We now analyze the relationship of two triangle circles with overlapped nodes as shown in Theorem 1.

**Theorem 1.** Given a cross edge e, for a destination D, let c₁ be the triangle circle constructed by e₁, and let the peak of c₁ be X. Given another cross edge e₂ for D, let c₂ be the triangle circle constructed by e₂ and let the peak of c₂ be Y. If c₁ and c₂ have overlaps, X must be equal to Y or X is the ancestor of Y, or Y is the ancestor of X on the routing tree of D.

**Proof.** We suppose that an overlapped edge of c₁ and c₂ is A. According to the definition of LCA, X=Primary₀(A) n₁=0, Y=Primary₀(A) m₂=0. If n₁=m, X=Y, otherwise if n₁>m, X=Primary_n₁(A) n₁=m₂=0, X is an ancestor of Y on the routing tree of D; otherwise n₁<m, Y=Primary_n₂(A) m₂=0, Y is an ancestor of X on the routing tree of D.

According to Theorem 1, if two triangle circles for a destination have overlapped nodes, the peaks of the two triangle circles must be the same or be ancestor-relationship, so they must be comparable. Therefore, we can summarize such a rule as follows.

**Rule 2.** Given a destination, if a router is contained in two or more triangle circles for a destination, the backup next hop of the router towards the destination should be assigned using the assigning rule according to the triangle circle with the highest peak. If there are two or more triangle circles with the same highest peak, any one of them is arbitrarily chosen.

Therefore, given a destination, the backup next hops of routers towards the destination on each triangle circle can be assigned using the assigning rule according to the sequence of the peaks of the triangle circles for the destination from high to low. If the backup next hop of a router towards a destination has been assigned, it should not be re-assigned, which assures that if a router is contained in two or more triangle circles with respect to a destination, the backup next hop of the router towards the destination will be the one assigned according to the highest triangle circle.

Given two triangle circles for a destination, C₁ and C₂, if the peak of C₁ is higher than C₂, the peak of C₁ must be earlier visited than the peak of C₂ in the depth-first search on the routing tree of the destination. Therefore, given a destination, we can assign the backup next hop to the destination for routers on each triangle circle according to the visit sequence of the peaks of the triangle circles in the depth-first search on the routing tree of the destination. The detailed algorithm in the centralized way is as shown in Fig. 5. The Assign function as shown in Fig. 6 is extracted from the RPFP-I algorithm in order to save space because it is frequently used. The reason for breaking the while loop in 1 in the Assign function is because of Theorem 2, reducing the computational complexity.

**Theorem 2.** In the Assign function, if the backup next hop of the visiting node Now towards D has been assigned, the backup next hops of all the nodes Primary₀(Now) 1 ≤ i ≤ m - 1, where Primary₀(Now) = C, towards D must also have been assigned.

---

**Algorithm 1 RPFP-I**

// Compute the backup next hop of each router towards each destination in the centralized way.

**Input:** The network graph \( G < V, E > \)

**Output:** Backup[S][D] denotes the backup next hop of router S towards D.

**Begin:**

1: for \( D = 1; D \leq |V|; D \leftarrow D + 1 \) do

2: **Step 1:** Calculate the routing tree of destination D and save to Primary[D]; \( Primary_D[S] \) denotes the primary next hop of router S towards D.

3: **Step 2:** Depth-first search on the routing tree of D, label all the routers orderly and the mappings from the DFS labels to the original IDs are saved in OID.; \( OID[L] \) denotes the original node ID labeled with L by DFS.

4: **Step 3:** Calculate the least common ancestor (LCA) for each cross link and save to LCA[...]; \( LCA[C] \) is the set of cross links whose Least common ancestor is router C.

5: **Step 4:** Assign the backup next hop towards D for each router as follows:[Line 6–12].

6: for \( L = 1; L \leq |V|; L \leftarrow L + 1 \) do

7: \( C \leftarrow OID[L] \);

8: for all Link A-B ∈ LCA[C] do

9: Assign(A, B, C, Primary_D, Backup);

10: Assign(A, B, C, Primary_D, Backup);

11: end for

12: end for

13: end for

---

**Function 1 Assign(A, B, C, D, Primary_D, Backup)**

// D is the destination. C is the LCA of A and B.

// Assign the backup next hop for D for each router from A to C.

**Begin:**

1: Last ← B; Now ← A;

2: while Now ≠ C do

3: if Backup[Now][D] has not been assigned then

4: Backup[Now][D] ← Last;

5: Last ← Now; Now ← Primary_D[Now];

6: else

7: break;//(1)

8: end if

9: end while

---
PROOF. As described in the assign function, the cross link is A-B. C is the LCA of A and B. Because the backup next hop of Now towards D has been assigned, Now must exist in another triangle circle for D. Let the peak of the triangle circle \( Y = \text{Primary}_D(\text{Now}) \). According to the RPFP-1 algorithm, \( Y \) must be C or be higher than C, so \( n \geq m \). All the nodes \( \text{Primary}_D(\text{Now}) \) \( 1 \leq j \leq n - 1 \) have been assigned. Therefore, this theorem is proven.

THEOREM 3. RPFP-1 provides complete protection against all the valid single link failures under the universal deployment.

PROOF. See Appendix A.

The decentralized RPFP-1 algorithm, which only works in the link-state-protocol-based network, only needs to make the following two modifications based on the centralized RPFP-1 algorithm. (a) Only the backup next hops of the router itself are computed rather than those of all the routers, which reduces the computational complexity on average, but does not reduce that at the worst. (b) If the visit sequence of two triangle circles for a destination ties according to the centralized RPFP-1 algorithm, the decentralized RPFP-1 algorithm should break the tie according to the router IDs in order to guarantee the consistency of all the routers.

According to the RPFP-1 algorithm, only one backup next hop for each destination is needed. We now analyze the complexity of the decentralized RPFP-1 algorithm. The routing tree of a destination can be constructed by collecting routing information from routers in the non-link-state-protocol-based network. In the link-state-protocol-based network, the routing tree of a destination is constructed by calculating the shortest path tree towards the destination. The complexity of calculating the shortest path tree towards a destination depends on the algorithm. If using the Dijkstra algorithm [21], the complexity is \( O(V^2) \). If using the Johnson algorithm [21], the complexity is \( O(E^2\log(E)) \). Because the network is a sparse graph, the Johnson algorithm runs faster than the Dijkstra algorithm. Thus, we use the Johnson algorithm. Therefore, the computational complexity of constructing the routing tree of a destination (Step 1) is \( O(E^2\log(E)) \). The computational complexity of a depth-first search on a routing tree (Step 2) is \( O(V) \). The computational complexity of calculating the LCA for all cross links for a destination using the Tarjan algorithm [21] (Step 3) is \( O(E) \). The computational complexity of Step 4 is \( O(E) \) because each cross link is visited once and each node is visited once. Therefore, if the backup next hop for all destinations are computed, the total computational complexity of the centralized RPFP-1 algorithm is \( O(V^2\log(E)) \). As shown in the centralized RPFP-1 algorithm, the space complexity of the centralized RPFP-1 algorithm is \( O(V^2) \). Similarly, the computational complexity of the decentralized RPFP-1 algorithm is still \( O(V^2\log(E)) \), but the space complexity the decentralized RPFP-1 is \( O(V+E) \) because only the backup routing table of the router itself needs to be saved.

D. The RPFP-2 algorithm

In this subsection, we will describe the RPFP-2 algorithm.

According to the RPFP-1 algorithm, if two or more triangle circles for a destination have the same high peak, the backup next hops of routers on those triangle circles towards the destination are assigned according to the arbitrary sequence of those triangle circles. If so, the protection against single link failures has no any problem, but the protection against single node failures may have problems as shown in Fig. 7. The destination is A. There are three cross links for A, which are E-C, I-J and J-K, constructing three triangle circles for A. The peak of the triangle circle constructed by E-C is the highest among the one of the three triangle circles, so the backup next hops of routers on the triangle circle constructed by E-C towards A are first assigned. The triangle circle constructed by I-J and the triangle circle constructed by J-K have the same high peak. If the backup next hops of routers on the triangle circle constructed by I-J towards A are earlier assigned than the ones of routers on the triangle circle constructed by J-K, as shown in Fig. 7 (a), the affected traffic from J towards A cannot be protected when Node B fails. Reversely, if the backup next hops of routers on the triangle circle constructed by J-K towards A are earlier assigned than the ones of routers on the triangle circle constructed by I-J, as shown in Fig. 7 (b), the affected traffic from J towards A can be protected when Node B fails. Note that according to RPFP-1, if the backup next hop of a router towards a destination has been assigned, the backup next hop of the router towards the destination will not be re-assigned. The difference between the two triangle circles constructed by I-J and J-K is that the triangle circle constructed by J-K has a non-peak router E, whose backup next hop towards A has been assigned, while the one constructed by I-J does not have such a router. Therefore, we can summarize such a rule as follows.

Rule 3. If two or more triangle circles for a destination have the same high peak, the triangle circle where there are at least one non-peak routers whose backup next hops towards the destination have been assigned should be earlier visited than those where there does not exist such a router.
Algorithm 2 RPFP-2
//Compute the backup next hop of each router towards each destination in the centralized way
Input: The network graph $G = (V, E)$;
Output: Backup path tree $\text{LCA}[i]$ for all $i$;
Begin:
  1. For $D - 1; D < |V|$, $D = D + 1$ do
     2. Step 1, 2, 3: The same to RPFP-1.
     3. Step 4: Assign the backup next hop towards D for each router as follows [Line 4–24].
     4. For $L = 1; L < |V|$, $L = L + 1$ do
        5. $C = \text{OD}(L)$;
        6. $\text{Flag} = \text{True}$;
        7. While $\text{Flag} = \text{True}$ do
           8. $\text{Flag} = \text{False}$;
           9. For all Link $A\rightarrow B \in \text{LCA}[i]$ do
              10. if $A\rightarrow B$ has not been visited and there exist assigned non-peak nodes in the triangle circle constructed by $A\rightarrow B$ then
                 11. Assign $(A, B, C, \text{Primary}, \text{Backup})$;
                 12. Assign $(B, A, C, \text{Primary}, \text{Backup})$;
                 13. $A\rightarrow B$ is marked visited; $\text{Flag} = \text{True}$;
                 14. Break;
              end if
           end for
        end while
     end for
end for

Fig. 8. The pseudo codes of the centralized RPFP-2 algorithm

Based on the centralized RPFP-1 algorithm, the centralized RPFP-2 algorithm only adds to use Rule 3 to determine the visit sequence of the triangle circles with the same high peak for a destination. The detailed RPFP-2 algorithm is as shown in Fig. 8. The decentralized RPFP-2 algorithm needs to make the same modifications based on the centralized RPFP-2 algorithm as the same as decentralized RPFP-1 algorithm do based on the centralized RPFP-1 algorithm.

THEOREM 4. RPFP-2 provides complete protection against all the valid single link and node failures under the universal deployment.

PROOF. See Appendix B.

We now analyze the complexity of the centralized RPFP-2 algorithm. The complexity of Step 1–3 of the centralized RPFP-2 algorithm is the same to that of centralized RPFP-1 algorithm. The complexity of Step 4 of the centralized RPFP-2 algorithm is as follows. We let $e_i$ be the number of links in $\text{LCA}[i]$ and $v_i$ be the number of visited nodes in the loop (Line 9–16) for $\text{LCA}[i]$. The complexity of Step 4 of the centralized RPFP-2 algorithm is:

$$\sum_{i=1}^{3} e_i \times v_i \leq \sum_{i=1}^{3} e_i \times \sum_{i=1}^{3} v_i = E \times V$$

Therefore, the computational complexity of Step 4 of the centralized RPFP-2 algorithm is $O(E^2 V)$. Therefore, the total computational complexity of the centralized RPFP-2 algorithm is $O(V \times E + V \times E \times \log(E))$. In general, $\log(E)$ is much smaller than $V$. Therefore, the total computational complexity of RPFP-2 is $O(V \times E)$. The space complexity of the centralized RPFP-2 algorithm is $O(V \times E)$. Similarly, the computational complexity of the decentralized RPFP-2 algorithm is $O(V \times E)$ and the space complexity of the decentralized RPFP-2 algorithm is $O(V \times E)$.

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>THEORETICAL COMPARISONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric a.1</td>
<td>complete</td>
</tr>
<tr>
<td>Metric a.2</td>
<td>incomplete</td>
</tr>
<tr>
<td>Metric b</td>
<td>One</td>
</tr>
<tr>
<td>Metric c.1</td>
<td>$O(V \times E \times \log(E))$</td>
</tr>
<tr>
<td>Metric c.2</td>
<td>$O(V \times E \times \log(E))$</td>
</tr>
<tr>
<td>Metric d.1</td>
<td>$O(V \times V \times V)$</td>
</tr>
<tr>
<td>Metric d.2</td>
<td>$O(V \times V \times E)$</td>
</tr>
</tbody>
</table>

TABLE II | TOPOLOGY INFORMATION |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>#nodes</td>
<td>12</td>
</tr>
<tr>
<td>#links</td>
<td>15</td>
</tr>
</tbody>
</table>

I. PERFORMANCE EVALUATIONS

In this section, we will evaluate RPFP-1 and RPFP-2 and compare them with the other no-tunnel-based IPFRR methods from the following four metrics: (a) the protection ratio under the universal deployment; (b) the memory cost of the forwarding table; (c) the computational complexity of algorithms; (d) the space complexity of algorithms. In Subsection A, we will evaluate them in theoretical analysis. In Subsection B, we simulated RPFP-1, RPFP-2 and the other no-tunnel-based IPFRR methods to verify their performance.

A. Theoretical analysis

The theoretical comparisons in the four metrics are as shown in Table I. We suppose that LFA and U-turn use the Dijkstra algorithm to calculate the shortest path tree because the OSPF protocol uses the Dijkstra algorithm in default. In Table I, the row header describes the IPFRR methods and the column header describes the metrics, which are described in detail as follows. Metric a.1 means whether complete link protection can be achieved under the universal deployment. Metric a.2 means whether complete node protection can be achieved under the universal deployment. Metric b refers to the number of backup next hops for one destination. Metric c.1 denotes the computational complexity in the decentralized way. Metric c.2 denotes the computational complexity in the centralized way. Metric d.1 denotes the space complexity in the decentralized way. Metric d.2 denotes the space complexity in the centralized way. The meanings of some words in the cells are introduced as follows. ‘One’ means only one backup next hop is needed. $V$ is the number of routers in the network and $E$ is the number of links in the network. The parameter $k$ denotes the number of its neighbors plus itself. The parameter $m$ denotes the number of its neighbors and its neighbors of neighbors plus itself. As shown in Table I, only RPFP-1 and RPFP-2 can provide complete link protection under the universal deployment. Only RPFP-2 can provide complete node protection under the universal deployment.
tion cannot be provided, we will give the exact values in different topologies in the following subsection. In terms of the number of the backup next hops, each of the above IPFRR methods only needs one backup next hop for one destination. In terms of computational complexity, RPFP-2 is the highest among them. In terms of space complexity, the above IPFRR methods are the same.

B. Simulation data and environments

We collected six intra-domain topologies with inferred link weights from Rocketfuel [22]. In addition, we collected a real backbone topology from Abilene [23]. The summary information of those topologies is shown in Table II. The row header describes the names of those topologies. In the column header, ‘#nodes’ denotes the number of nodes and ‘#links’ denotes the number of links.

For each IPFRR method, we simulated to generate the forwarding table including the primary next hops and the backup next hops for each router, and we then simulated the forwarding process for each valid single link failure case and valid single node failure case, and calculated the link protection ratios and the node protection ratios for those no-tunnel-based IPFRR methods in those topologies. The simulations were run on a 2.93GHz processor with 4G memory space. Table III shows the link protection ratios of those IPFRR methods in different topologies under the universal deployment. Table IV shows the node protection ratios of those IPFRR methods in different topologies under the universal deployment. Now we explain the meanings of the row headers and the column headers for Table III and Table IV. The row headers describe the different topologies.

As shown in Table III, the link protection ratio provided by both RPFP-1 and RPFP-2 is definitely 100%, which validates our theoretical proof for that. The link protection ratio provided by LFA is 67% on average. The link protection ratio provided by U-turn is 90% on average. As shown in Table IV, the node protection ratio provided by RPFP-2 is definitely 100%, which validates our theoretical proof for that. Surprisingly, the node protection ratio provided by RPFP-1 is 99.90% on average. The node protection ratio provided by LFA is 71% on average. The node protection ratio provided by U-turn is 91% on average.

II. PARTIAL DEPLOYMENT ISSUE

In this section, we address the partial deployment issue. We describe how RPFP-1 (or RPFP-2) works in an intra-domain, where only a part of routers are upgraded to support RPFP-1 (or RPFP-2). In addition, we design a nearly optimal strategy of upgrading routers. Moreover, we compare RPFP-1 and RPFP-2 with LFA and U-turn in terms of the protection ratio under partial deployment. Given a protection path, the routers on the normal forwarding sub-paths of the protection path do not need to be upgraded. Therefore, even if not all routers are upgraded, a proportion of failure cases can be protected. To support the partial deployment, only the assign function needs to be modified as follows: if the current visiting node does not support RPFP, break the while loop.

Now we introduce how to upgrade routers optimally. Given the number of upgraded routers \( n \), choosing which \( n \) routers should be upgraded to maximize the protection ratio is a NP-hard problem. Thus, we need to design a heuristic strategy to determine the nearly optimal upgrade arrangement. The optimizing objective is to maximize the protection ratio. However, the maximum link protection ratio and the maximum node protection ratio cannot be achieved simultaneously. According to [20], 70% of network failures are single link failures, so we
use the maximum link protection ratio as the optimizing objective. Given the number of upgraded routers $n$, our heuristic upgrade strategy is to one by one choose the router that makes the protection ratio increase most if the router is upgraded.

Using that upgrade strategy, we give the increase curves of the link protection ratios as the upgrade ratios increase for RPFP-1 in different topologies in Fig. 9. Those increase curves of the link protection ratios of RPFP-2 and the node protections of RPFP-1 and RPFP-2 are nearly close to those in Fig. 9, so we omit them for space restriction. We find that if only 40% of routers are upgraded to support RPFP-1 or RPFP-2, more than 80% of valid single failure cases will be protected. Of course, if 100% routers are upgraded to support RPFP-2, 100% of valid single failure cases will be protected. In addition, we generated the upgrade arrangements for LFA and U-turn using that upgrade strategy, and compared RPFP with them in terms of the protection ratio under partial deployment. The results in the topology of AS1221 are as shown in Fig. 10. The results in other topologies show the same findings to those in Fig. 10, so we omit them due to space restriction. We find that RPFP has much higher protection ratio than LFA and U-turn under any partial deployment ratio. Especially, LFA and U-turn still cannot provide complete protection against all the valid single failures even under the universal deployment.

III. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a new kind of no-tunnel-based IPFRR method, named with RPFP, which can provide complete protection against all the valid single failures under the universal deployment. In addition, only 40% of routers are upgraded to support RPFP, more than 80% of valid single failure cases will be protected. RPFP performs significantly better with acceptable higher computation cost than the other existing no-tunnel-based IPFRR methods on the failure recovery.

RPFP is a great advance in IPFRR, although it still has such a limitation, which ubiquitously exists in all the IPFRR methods. The limitation is that if another failure or severer congestion happens to occur on the repair path, through which the affected traffic is rerouted, the affected traffic may still be dropped or delayed. Reversely, if the repair path has neither any failure nor congestion, no matter how many hops the repair path has, the affected traffic must be able to be fast rerouted to their destination. It is because the bandwidth of each link is at least 100Mb/s, even 1G/s or 10G/s, the transfer delay on a path with so wide bandwidth and without congestion is negligible. Therefore, in the future work, we will expand RPFP to conquer that challenge, i.e. how to again reroute the affected traffic if another failure or congestion occurs on the repair path. According to the mechanism of RPFP, RPFP has a great potential to solve that problem.

ACKNOWLEDGEMENT

This research is supported by the National High-tech R&D Program ("863" Program) of China (No.2013AA010605), the National Science Foundation of China (No.61073172), and National Key Basic Research Program ("973" Program) of China (No.2009CB320501). In addition, we would like to thank Yang Yu for some experiments, and also thank Yang-yang Wang for some very good comments on this work. Jun Bi is the corresponding author.

REFERENCES

[22] The router-level ISP topologies from rocketfuel project: http://www.cs.washington.edu/research/networking/rocketfuel

APPENDIX

In this section, we will prove Theorem 3 and Theorem 4. Before proving the two theorems, we prove some lemmas respectively. In this appendix, the tree link, cross link and the triangle circle are with respect to the given destination in.
lemmas or theorems for simplicity.

A. Proof for Theorem 3

**Lemma 1.** Given any destination D and any link L, if L is not a bridge, L must be contained in at least one triangle circles on the routing tree of D.

**Proof.** To prove that L exists in a triangle circle on the routing tree of D, we only need to prove that L exists a circle with only one cross edge of L. Let L be denoted by S-P. Because L is not a bridge, L must exist in a circle CIRCLE denoted by \{V_i - V_{i+1} \ldots - V_n - V_1\}, where \(V_1 = S\) and \(V_n = P\). There must exist at least one cross edges on CIRCLE, otherwise, CIRCLE will be completely composed of tree edges, which contradicts with the characteristic of tree. We use \{V_{b_1} - V_{b_1+1}, V_{b_2} - V_{b_2+1}, \ldots, V_{b_m} - V_{b_{m+1}}\}, whose sequence is consistent with CIRCLE, to denote the set of cross edges on CIRCLE, where \(V_{b_k}\) is P or connects to P not via S through tree edges and \(V_{b_{m+1}}\) is S or connects to S not via P through tree edges.

**Initialization.** \(V_{b_1}\) is P or connects to P not via S through tree edges.

**Iterations:** \(1 \leq i \leq m - 1\). Firstly, the statement that \(V_{b_i} = P\) or connects to P not via S through tree edges holds. Secondly, if \(V_{b_{i+1}}\) is S or connects to S not via P through tree edges, then link S-P exists a circle with only one cross edge \(V_{b_{i+1}} - V_{b_i}\), so this lemma is proven; otherwise, \(V_{b_{i+1}}\) must be P or connect to P not via S through tree edges, then the next iteration begins with \(i = i + 1\).

**Termination:** \(i = m\). Because \(V_{b_{m+1}}\) is S or connects to S not via P through tree edges, Link S-P exists a circle with only one cross edge \(V_{b_m} - V_{m+1}\). This lemma is proven.

Therefore, this lemma may be proven in iterations. At least, this lemma must be able to be proven in the termination.

**Theorem 3.** RPFP-1 provides complete protection against all the valid single link failures. Now we prove that RPFP-1 can also provide complete protection against all the valid single node failures. For each valid node failure case \((S,D,M)\), where S is the source, D is the destination and M is the node that fails. Let L be denoted by W-U, where Primary\(_D\) (W) = U, according to the definition of the valid single failure case, S must be the descendant of U, and S can connect to D after L is removed. Therefore, L is not a bridge, according to Lemma 1, L must be contained in at least one triangle circles on the routing tree of D. According to the RPFP-1 algorithm and the RPFP forwarding logic, (W,D,L) can be protected by RPFP-1. Because S is the descendant of W, (S,D,L) can also be protected by RPFP-1. This theorem is proven.

B. Proof for Theorem 4

**Lemma 2.** Given any destination D and any other node M, suppose that S is a son of M and P is the father of M on the routing tree of D, if S-M-P is contained in a circle, named with CIRCLE, S-M must be contained in a such circle, on which there must exist at least a cross link whose LCA's are higher than M (Requirement 1) and the LCA of all the other cross links if they exist is M (Requirement 2).

**Proof.** We suppose that CIRCLE is denoted by \{ V_1 - V_2 - \ldots - V_n - V_1 \}, where \(V_1 = P, V_2 = M\) and \(V_3 = S\). There must exist at least one cross edges on CIRCLE, otherwise, CIRCLE will be completely composed of tree edges, which contradicts with the characteristic of tree. We use \{V_{b_1} - V_{b_1+1}, V_{b_2} - V_{b_2+1}, \ldots, V_{b_m} - V_{b_{m+1}}\}, whose sequence is consistent with CIRCLE, to denote the set of cross edges on CIRCLE, where \(V_{b_1}\) is S or connects to S not via M through tree edges and \(V_{b_{m+1}}\) is P or connects to P not via M through tree edges.

**Initialization.** \(V_{b_1}\) connects to M not via P through tree edges.

**Iterations:** \(1 \leq i \leq m - 1\). Firstly, the statement that \(V_{b_i}\) connects to M not via P through tree edges holds. \(V_{b_{i+1}}\) must be one of the following two conditions:

(a) \(V_{b_{i+1}} = P\) or connects to P not via M through tree edges, then \(V_{b_i} - V_{b_{i+1}}\) is contained in a triangle circle, whose peak is higher than M, so Requirement 1 is satisfied.

(b) \(V_{b_{i+1}}\) connects to M not via P through tree edges, \(V_{b_i} - V_{b_{i+1}}\) must be one of the following two conditions:

(b.1) The LCA of \(V_{b_i} - V_{b_{i+1}}\) is M, which satisfies Requirement 2.

(b.2) The LCA of \(V_{b_i} - V_{b_{i+1}}\) is S, or lower than S, then S can reach \(V_{b_{i+1}}\) through tree edges, so \(V_{b_i} - V_{b_{i+1}}\) can be eliminated to satisfy Requirement 2.

The next iteration begins with \(i = i + 1\).

**Termination:** \(i = m\). Because \(V_{b_{m+1}}\) is P or connects to P not via M through tree edges, then \(V_{b_m} - V_{b_{m+1}}\) is contained in a triangle circle, whose peak is higher than M, so Requirement 1 is satisfied.

Therefore, Requirement 1 and Requirement 2 must be able to be satisfied. This theorem is proven.

**Theorem 4.** RPFP-2 provides complete protection for all the valid single link and node failures under the universal deployment.

**Proof.** Obviously, RPFP-2 is a special case of RPFP-1. Therefore, RPFP-2 can provide complete protection against all the valid single link failures. Now we prove that RPFP-2 can also provide complete protection against all the valid single node failures as follows. For each valid node failure case \((S,D,M)\), where S is the source, D is the destination and M is the node that fails. According to the definition of the valid single failure case, S must be the descendant of M and S can connect to D after M is removed. If S is not a son of M, we suppose that S reaches M via U that is a son of M. Because S can connect to D after M is removed. U can also connect to D after M is removed. We suppose that the father of M is P, then U-M-P is contained in a circle. According Lemma 2, \(U-M\) must exist in such a circle, on which there exists a cross link whose LCA is higher than M and the LCA of all the other cross links if this is higher than M. Therefore, according to the RPFP-2 algorithm and the RPFP forwarding logic, (U,D,M) can be protected by RPFP-2 when node M fails. Therefore, (S,D,M) can also be protected by RPFP-2 when node M fails. If S is a son of M, obviously, (S,D,M) can also be protected by RPFP-2 when node M fails. This theorem is proven.