Increased Robustness with Interface Based Permutation Routing

Hung Quoc Vo
Simula Research Laboratory

Olav Lysne
Simula Research Laboratory

Amund Kvalbein
Simula Research Laboratory

***Abstract*—A prime objective of fault tolerant routing methods is the availability of multiple routing options at each hop. The methods that are currently implemented in IP networks, such as Equal-Cost Multi-Path (ECMP) and Loop Free Alternates (LFA), share the following four properties: First, they work with a hop-by-hop forwarding strategy optimized for the fault free case. Second, they do not require information associated with network faults included in the packet header. Third, they do not form forwarding loops, even under multiple failures in the network. Finally, they are compatible with standard link state routing protocols. However, ECMP and LFA give very poor fault coverage; in most cases fewer than 50% of primary next-hops are protected when using typical link weight settings. This paper presents a new routing method that combines the concept of permutation routing with interface-specific forwarding. Our method results in a routing strategy that strictly adheres to the four stated design properties. Through experiments we show that we protect more than 97% of the primary next-hops for all tested ISP networks.**

I. Introduction

IP networks are the preferred transportation medium for time-critical services such as online trading, remote monitor and control systems, telephony and video conferencing. Fast recovery from network failures has therefore become an important requirement when designing networks. Currently implemented solutions to the fast reroute problem in IP networks are Equal-Cost Multi-Path (ECMP) and Loop Free Alternate (LFA) [1], both of which share four common properties that are critical for adoption: First, they do not require the network operator to change the traditional hop-by-hop forwarding strategy that is optimized for the fault free case. Second, they do not require addition of fault-information included in the packet header. Third, they do not suffer from routing loops, even when there are multiple faulty components in the network. Finally, they work with the existing standard link state routing protocols such as OSPF or IS-IS. ECMP and LFA are simple but give limited protection against network failures; in most cases they protect below 50% of primary next-hops when using typical link weight settings [2].

Maximizing failure coverage has become a common routing objective for increased robustness in many recently proposed routing schemes. Several solutions, specifically Tunnels [3], Not-via [4], MRC [5] and IDAGs [6], promise full coverage for single network component faults with the cost of altering the traditional IP forwarding. More related to our work, Failure Insensitive Routing (FIR) [7] introduces the interface-specific forwarding concept, which has potential to be deployed with low complexity on modern high-performance routers. FIR also offers full fault coverage for single link faults but it may cause routing loops when there exist multiple failures in the network [8].

In this paper, we propose interface based permutation routing, which combines the concept of permutation routing [9] with interface-specific forwarding [7]. The method aims to *maximize* protection coverage for IP networks by giving multiple loop-free next-hops towards a destination for each incoming interface of a router. Importantly, our routing method shares with ECMP and LFA the four listed properties.

Permutation routing [9] is a new and flexible approach for calculating multiple loop-free next-hops in networks with the traditional hop-by-hop forwarding. Permutation routing is based on the fact that any routing strategy can be expressed as a *permutation* (sequence) of nodes that are involved in traffic forwarding to a destination. The routing is loop-free if packets are forwarded in one direction towards the destination regarding to node ordering in the permutation. Permutation routing only takes the topology information that is collected by link state routing protocols as the input for its construction, and hence no new control plane signaling is needed.

The main focus of this paper is to construct permutation routing of *interfaces* that approximately maximizes the number of primary interface-destination pairs that have more than one available next-hop. Importantly, our routing method can easily be integrated with existing link state routing protocols, OSPF or IS-IS, and can be used to augment shortest path routing tables with additional forwarding entries. We show that our interface based permutation routing can protect more than 97% of the primary next-hops in all tested ISP topologies. This is significantly better than LFA and ANHOR-SP [9], which protect from 21% to 67% and from 29% to 81%, respectively, in the topologies we study. Note that ANHOR-SP is a routing strategy constructed by using the concept of permutation routing, which seeks to maximize protection coverage for traditional IP networks. In this paper we only focus on finding a set of loop-free paths, and leave the important topic of load-balancing across these paths for future work.

The rest of the paper is structured as follows. Section II explains why the interface-specific forwarding is beneficial over the traditional IP forwarding and proposes iNHOR as our main routing objective. Section III describes in detail a heuristic to create interface based permutation routings that approximate iNHORs. Section IV assesses the multipath

capability of our routing method. We review related work in Section V and conclude the paper in Section VI.

II. Interface Next-Hop Optimal Routing

This section first reviews the interface-specific forwarding concept and explains by examples its ability to increase routing robustness for IP networks. We then introduce Interface Next-Hop Optimal Routing (iNHOR) as our main objective function for a robust routing.

A. Link-specific forwarding

In IP networks with interface-specific forwarding, packets are routed based on both their incoming interfaces and their destination addresses. For that reason, each *line-card* of a router will maintain its own *distinct* forwarding table that maps each network destination to eligible outgoing interfaces. Fortunately, such design is available in most modern routers, in which forwarding engines are integrated into line cards in a distributed mode [10]. We explain by examples how interface based routing can help increase robustness over the traditional IP routing.

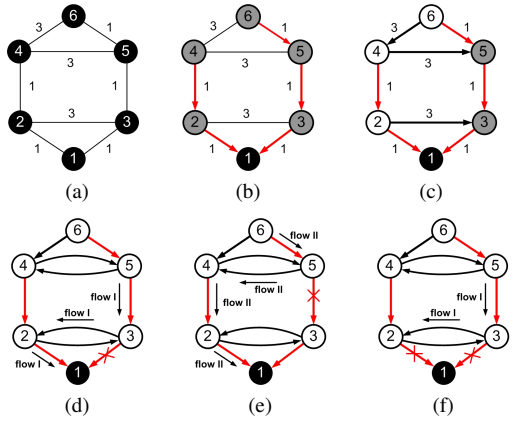


Fig. 1: (a) A topology. (b) An SPT rooted at node 1. (c) A DAG, which contains the SPT, rooted at node 1. Fast rerouting under interface-specific forwarding if (d) link $(1,3)$ fails, (e) link $(3,5)$ fails, (f) multiple links fail.

Fig. 1a illustrates a simple topology with six nodes and eight (bidirectional) links with their corresponding link weights. Fig. 1b is a shortest path tree (SPT) rooted at node 1. We say that a node is *protected* if it has *at least* two next-hops, no node in Fig. 1b is protected (gray shadowed nodes). Other routing methods, e.g. ANHOR-SP [9], can improve the protection coverage by adding *non-shortest* path branches to form a better-connected routing graph, mandatorily a directed acyclic graph (DAG). Fig. 1c is such an instance. Obviously, Fig. 1c is the most robust loop-free routing graph rooted at node 1 that the traditional IP routing can provide because adding any more directed link will cause loops. Consequently, we fail in protecting node 3 and node 5 under single failures.

However, node 3 and node 5 of the topology can be protected if interface-specific forwarding is used. Correspondingly, node 3 (in Fig. 1d) will reroute flow I to node 2 under failure of link $(1,3)$. Upon receipt of rerouting packets from incoming interface $(3 \rightarrow 2)$, node 2 will forward them to destination 1. Likewise, node 5 (in Fig. 1e) will safely reroute flow II to node 4 under failure of link $(3,5)$ where it is then forwarded to node 2 without causing loops. To do that, *all interfaces* of nodes should be installed with loop-free forwarding tables towards destination 1. In those tables, next-hops marked with *primary* are used if available. Other next-hops are called secondaries and only used when all primaries are not in service. If there does not exist any next-hop (either primary or secondary) for a specific incoming interface, packets will be *discarded* at that interface. As shown in Fig. 1f, flow I is dropped at interface $(3 \rightarrow 2)$ due to failures of both links $(1,3)$ and $(1,2)$. We will show how to construct such a forwarding table for each interface in Section III, but we first introduce our main routing objective.

B. Interface Next-Hop Optimal Routing

For maximizing fault tolerance and load-balancing capabilities of a network with interface-specific forwarding, it is important that the routing offers more than one available next-hop for as many primary incoming interface-destination pairs as possible. This leads us to the following optimization criterion for Interface Next-hop Optimal Routing (iNHOR):

Definition 1: Given a routing for a destination, an incoming interface on the SPT is called a primary interface (pI) and an incoming interface *not* on the SPT a secondary interface. An iNHOR is an interface based routing that contains the SPT and maximizes the number of primary interface-destination (pI-D) pairs with *at least* two next-hops.

By maximizing fault tolerance capability for primary interfaces, iNHOR has a great potential to increase robustness for IP networks under failures. In addition, the SPT inclusion constraint of iNHOR is usually required for traffic engineering purpose, e.g. shortest paths likely provide low transmission latency for voice or video traffic. In the next section, we introduce a method that helps construct a loop-free forwarding table for each interface using permutation routing.

III. Algorithm Design

We design a heuristic for generating interface based permutation routings for a given topology that approximate iNHOR. We call the resulting routing Approximate iNHOR (AiNHOR). The heuristic is based on the generic framework introduced in [9] with two modifications. First, the given topology is transformed into a line graph [11] where each vertex [1] represents a directed link of the topology. Second, a new selection strategy is proposed to pick a node from the resulting line graph for each position in the permutation routing which represents an AiNHOR. We first review how to construct a permutation routing in general.

[1] we use "node" and "link" for the connectivity of given input topologies and "vertex" and "edge" for their corresponding line graphs.

A. Permutation Routing

We model the given topology as graph $G = (V(G), E(G))$ where $V(G)$ is the set of nodes and $E(G)$ is the set of directed links of G[2]. Let p and q denote the cardinalities of $V(G)$ and $E(G)$, respectively. Given G and the constraint function C(u), which is defined to realize a selected routing objective, on each node, a permutation routing for destination d is:

1) A sequence of N nodes, called a permutation which is denoted by P, whose ordering satisfies C(u). Destination d is at the left-most position of P.
2) A node can forward its packets to all its neighboring nodes that occur before it in P.

We use the well-known backtracking algorithm [12] to create such permutation P. The key procedure of the algorithm is to assign a node to a variable that represents a position in the permutation so that the assignment satisfies C(u). Fig. 2 illustrates the basic assignment procedure for variable p_{i+1} in which two key functions `Update` and `Select` work as filters to control the assignment. Specifically, we have a set of N variables, $P = \{p_1, p_2, \ldots, p_N\}$, in a fixed order from p_1 to p_N. Function `Update` is designed to generate domain D_{i+1} for variable p_{i+1}. We refer to D_{i+1} as the *candidate set* which consists of nodes that can be assigned to variable p_{i+1}. Then function `Select` will pick one node in D_{i+1} that fulfills C(u) and assign it to variable p_{i+1}. In the figure, each pair $\langle p_i, u_i \rangle$ represents the assignment of the node u_i to variable p_i. The assignment of nodes to a subset of variables $\{p_1, p_2, \ldots, p_i\} \subseteq P$ given by $\{\langle p_1, u_1 \rangle, \ldots, \langle p_i, u_i \rangle\}$ is called a *partial* permutation with i nodes. For simplicity, we abbreviate it to $\vec{p_i}$.

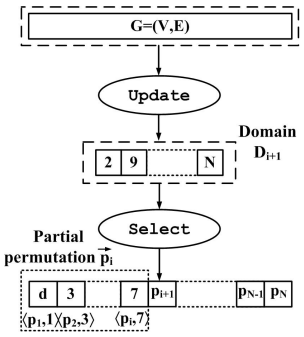


Fig. 2: Basic assignment procedure for variable p_{i+1}

B. Line Graphs

ANHOR (Approximate Next-Hop Optimal Routing) and ANHOR-SP (Shortest Path compatible ANHOR) are two routing strategies which are computed using permutation routings of nodes [9]. However, in this paper we desire to use permutation routing to calculate next-hops for each interface (not for each node). In other words, the permutation routing in this case should be an ordering of all directed links (incoming interfaces) of the given topology. Fortunately, we can achieve such a permutation routing without modifying the described framework by using the line graph of the topology as the input.

[2]each link in the topology is modeled as two directed links.

Definition 2: Given a graph G, its line graph $\mathbf{G}$[3] $= (V(\mathbf{G}), E(\mathbf{G}))$ is a graph such that:

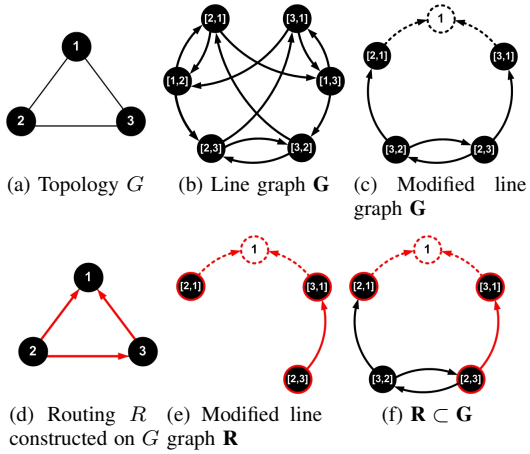
1) Each vertex of $\mathbf{G}$ represents a directed link of G; and
2) Two vertices $\mathbf{u}$ and $\mathbf{v}$ of $\mathbf{G}$ form a directed edge $(\mathbf{u} \rightarrow \mathbf{v})$ if and only if their corresponding directed links $(u_1 \rightarrow u_2)$ and $(v_1 \rightarrow v_2)$ of G are adjacent in G or $u_2 \equiv v_1$.

Following the definition of line graphs, we have propositions regarding to the connectivity, subset relationship and acyclic property of line graphs (proofs can be found in [11] or our technical report in [12]).

Proposition 1: Let G_1 and G be two directed graphs,

1) If G is connected, then $\mathbf{G}$ is connected.
2) If $G_1 \subset G$, then $\mathbf{G}_1 \subset \mathbf{G}$.
3) If G_1 is a DAG, then $\mathbf{G}_1$ is also a DAG.

From the definition of line graphs, we might think that $\mathbf{G}$ is a proper input for calculating permutation routing of interfaces. It is, however, infeasible since $\mathbf{G}$ does not include any vertex that corresponds to the real destination node. For the routing computation purpose, we should modify $\mathbf{G}$ by adding a virtual vertex that represents the destination and taking away vertices and directed edges that will not contribute to routing towards the virtual destination. For simplicity, we also use $\mathbf{G}$ to denote the modified version of $\mathbf{G}$. Fig. 3 illustrates the line graph $\mathbf{G}$ (Fig. 3b) and its modified version with an added virtual vertex and two added virtual directed edges in dotted lines (Fig. 3c) of the topology G in Fig. 3a.



(a) Topology G (b) Line graph $\mathbf{G}$ (c) Modified line graph $\mathbf{G}$

(d) Routing R constructed on G (e) Modified line graph $\mathbf{R}$ (f) $\mathbf{R} \subset \mathbf{G}$

Fig. 3: The topology, routing graph and their line graphs. Note that we denote $[u, v]$ by the vertex of $\mathbf{G}$, which is directed link $(u \rightarrow v)$ in G.

Let R be the SPT towards destination d which is constructed on G, its line graph with virtual destination d will be $\mathbf{R}$. According to Proposition 1, $\mathbf{R}$ is connected, $\mathbf{R} \subset \mathbf{G}$ and $\mathbf{R}$ is a DAG towards d. Fig. 3e and 3f illustrate those properties with a simple example of shortest path routing R in Fig. 3d. Note that line graph $\mathbf{G}$ in Fig. 3f has two types of vertices. There are vertices which are both on the line graphs $\mathbf{G}$ and $\mathbf{R}$ such as vertices 1, $[2, 1]$, $[3, 1]$ and $[2, 3]$ and one vertex, $[3, 2]$,

[3]We use boldface letters to denote line graphs and their objects, e.g. vertices and edges.

which is not on **R**. Since we desire that AiNHOR calculated from **G** is compatible with the shortest path routing, AiNHOR must include **R**.

C. Domain generation

With given line graphs **G** and **R**, we then define the domain for each variable. Because a vertex in any valid routing must have at least *one* next-hop towards the destination, the domain D_{i+1} for variable p_{i+1} can be the set that contains all vertices with at least one out-neighbor, which has been placed in partial permutation $\vec{p}_i$, as follow:

$$D_{i+1} = D_i \cup \{\ \mathbf{v} \in \boldsymbol{\mathcal{V}}_i \mid (\mathbf{v} \to \mathbf{u}) \in E(\mathbf{G}\} \setminus \{\mathbf{u}\} \quad (1)$$

where $\mathbf{u}$ is the vertex that has been assigned to variable p_i in the i-th assignment and $\boldsymbol{\mathcal{V}}_i$ is a set of vertices of **G**, excluding all vertices in $\vec{p}_i$ and D_i.

Domain D_{i+1} is usually large when i increases. We will divide D_{i+1} into smaller domains, on which our search would be more efficient. We have noticed that D_{i+1} may contain *three* types of vertices. First, they are vertices in **R** which have *all* their next-hops in **R** already placed in $\vec{p}_i$, denoted by D^a_{i+1}. Second, they are vertices which are *not* in **R**, denoted by D^b_{i+1}. Third, they are vertices in **R** which *do not* have *all* their next-hops in **R** already placed in $\vec{p}_i$, denoted by D^c_{i+1}.

For a vertex $\mathbf{v}$, let $c_{sp}[\mathbf{v}]$ be the number of next-hops in **R** placed in $\vec{p}_i$ and $n_{sp}[\mathbf{v}]$ be the total number of next-hops in **R**, the sub-domain D^a_{i+1} for variable p_{i+1} follows the recursive relation:

$$D^a_{i+1} = D^a_i \cup \{\mathbf{v} \in D_{i+1} \mid c_{sp}[\mathbf{v}] = n_{sp}[\mathbf{v}]\} \quad (2)$$

and sub-domain D^b_{i+1} is calculated as follow:

$$D^b_{i+1} = D^b_i \cup \{\mathbf{v} \in D_{i+1} \mid \mathbf{v} \notin \mathbf{R}\} \quad (3)$$

We then design a selection strategy to pick a vertex from defined sub-domains for a variable in each assignment to produce a permutation routing representing AiNHOR.

D. Selection strategy

We denote $\boldsymbol{\mathcal{R}}_i = (V(\vec{p}_i), E(\vec{p}_i))$ by the routing sub-graph towards to destination d which is represented by partial permutation $\vec{p}_i$. To achieve a robust routing from **G** while containing **R**, the vertex selected from sub-domain D^*_{i+1} (e.g. D^a_{i+1} or D^b_{i+1}) for variable p_{i+1} to form partial permutation $\vec{p}_{i+1}$ should be the vertex with the maximum number of out-neighbors already placed in $\vec{p}_i$:

$$|E(\vec{p}_{i+1})| = \max_{\forall \mathbf{u} \in D^*_{i+1}} |E(\vec{p}_i, \langle p_{i+1}, \mathbf{u}\rangle)| \quad (4)$$

and if $\mathbf{R}_{i+1}$ is a subset of **R** after $(i+1)$-th assignment, then

$$\mathbf{R}_{i+1} \subseteq \boldsymbol{\mathcal{R}}_{i+1} \quad (5)$$

We derive constraint function $C(\mathbf{u})$ to realize expression (4) and (5) as follow:

$$C(\mathbf{u}) = \begin{cases} \textbf{True} & \text{if } c[\mathbf{u}] = \max_{\forall \mathbf{v} \in D^*_{i+1}} c[\mathbf{v}] \\ & \text{and } \mathbf{R}_{i+1} \subseteq \boldsymbol{\mathcal{R}}_{i+1} \\ \textbf{False} & \text{otherwise} \end{cases}$$

where $c[\mathbf{u}]$ denotes the number of outgoing edges from $\mathbf{u}$ to $\vec{p}_i$ if $\mathbf{u}$ is selected in $(i+1)$-th assignment.

We then select one vertex from one of three sub-domains that satisfies $C(\mathbf{u})$ and place it in the permutation. We have two strategies: *first* choosing a vertex in D^a_{i+1} that satisfies (4) if it is not empty or *first* choosing a vertex in D^b_{i+1} that satisfies (4) if it is not empty. It is easy to verify that the resulting permutation routings in both cases will satisfy (5). However, the latter will increase the protection coverage because it places in the permutation *all* possible secondary links, which can become next-hops for primary links, before primary links. Fig. 4 summarizes the selection procedure for $(i+1)$-th assignment, in which we go through D^b_{i+1} first.

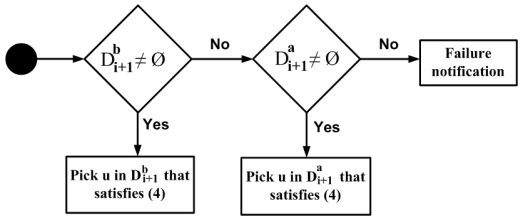


Fig. 4: The selection procedure for variable p_{i+1}

We apply our described algorithm to the topology in Fig. 1a and yield the permutation of 15 vertices towards destination 1: {**1**, **[2,1]**, **[3,1]**, [3, 2], [2, 3], **[4,2]**, **[5,3]**, [2, 4], [5, 4], [3, 5], [4, 5], [6, 4], **[6,5]**, [4, 6], [5, 6]}. In the permutation, those boldface vertices are the directed links of the topology that are on the shortest path towards node 1 (Fig. 1b). Forwarding tables for interfaces then are generated by allowing each vertex in the permutation to forward packets to all its neighboring vertices that occur before it in the permutation. Fig. 5 shows two examples of forwarding tables for interfaces of node 2 and node 3, extracted from the permutation routing of vertices.

Node	Incoming interface	Next-hop	Primary
2	4 → 2	1	Yes
		3	No
	3 → 2	1	Yes

Node	Incoming interface	Next-hop	Primary
3	5 → 3	1	Yes
		2	No
	2 → 3	1	Yes
		2	No

Fig. 5: Forwarding tables of interfaces of node 2 and node 3.

Proposition 2: Our selection procedure in Fig. 4 gives a backtrack-free algorithm for all connected directed graph.

Proof: We observe that whenever one of two domains D^b_{i+1} and D^a_{i+1} is not empty, we could find a vertex that satisfies (4). Therefore, our algorithm will perform backtracking step only if both D^b_{i+1} and D^a_{i+1} are empty at the same time in some iteration. However, that can never happen before all vertices have been placed in the permutation. If this is the case, there exists only candidate vertices in D^c_{i+1}. In other words, we can follow **R** from any vertex that has not been placed and always find a next-hop vertex that is not placed in $\vec{p}_i$. But this is impossible: since **R** is a DAG (Proposition 1), any path along the shortest path will eventually reach the destination, which is the first vertex placed in the permutation. ■

Due to the property of backtrack-freeness, with sparse topologies the computational complexity to construct one permutation towards *one* destination would be $O(|E(\mathbf{G})|+q\times|D|)$ where $|D|$ denotes the average size of domains D_i and $|E(\mathbf{G})|$ can be computed based on G that we may find in [11] or in our technical report [12]. In dense topologies, the total complexity of calculating permutations for *all* destinations can approach $O(q^3)$.

IV. Performance Analysis

We assess routing robustness of AiNHOR in terms of the improved number of next-hops for primary incoming interfaces towards all destinations. Since adopting secondary interfaces for packet transportation can lead to path inflation, we also investigate the hop-count length distribution with various allowed number of next-hops.

A. Evaluation Setup

We select six representative network topologies from the Rocketfuel project [13] for our evaluations. For each topology, we remove all nodes that will not contribute on routing (e.g. single out-degree node). The refined topologies are bi-connected graphs, listed in Table I in increasing order of their average node out-degrees. In addition, Table II shows their corresponding line graphs in increasing order of their average vertex out-degrees.

ECMP and LFA base their path calculation on link weights. To obtain realistic link weight settings, we implement local search heuristic [14] to optimize link load objective function under traffic matrix generated by the gravity model [15]. For AS1239, we, however, use unit link weights because the local search heuristic does not scale to a topology of this size.

TABLE I: Network topologies

AS	Name	Nodes	Links	Avg. Degree
1221	Telstra(au)	50	194	3.88
3967	Exodus(us)	72	280	3.89
1755	Ebone(eu)	75	298	4.00
3257	Tiscali(eu)	115	564	4.90
6461	Abovenet(us)	129	726	5.60
1239	Sprint(us)	284	1882	6.62

TABLE II: Line graphs

AS	Name	Nodes	Links	Avg. Degree
1755	Ebone(eu)	298	1432	4.80
3967	Exodus(us)	280	1372	4.90
1221	Telstra(au)	194	1030	5.30
6461	Abovenet(us)	726	5434	7.50
3257	Tiscali(eu)	564	4378	7.76
1239	Sprint(us)	1882	25988	13.81

B. Comparison

We compare our AiNHOR with shortest path based routings, ECMP and LFA, and recent solution ANHOR-SP. Comparisons to other interface based routing methods (e.g. FIR [7], NISR[16] and ESCAPE [17]) are less relevant because they do not fulfill the four properties stated in our introduction.

We evaluate robustness of all mentioned routing methods in terms of links, instead of nodes. Accordingly, for given incoming interface $(* \rightarrow i)$, j is called the primary next-hop of $(* \rightarrow i)$ if $(i \rightarrow j)$ is on the SPT towards d. Otherwise, j is called secondary next-hop and $(i \rightarrow j)$ is a secondary link towards d.

C. Robustness Evaluation

Fig. 6 shows fractions of primary interfaces with at least two next-hops under four methods. We observe that the fractions of AiNHOR vary slightly across six topologies and are above 97%. Especially, AiNHOR achieves 100% primary interfaces with two next-hops in AS6461 and AS3257. The good results come from properties of interface based routing and its operation on the bi-connected graph. That is a node with at least two outgoing interfaces may have the chance to installed two loop-free next-hops.

Obviously, AiNHOR gives a clear improvement over LFA and ANHOR-SP. Fig. 6 shows that LFA and ANHOR-SP protect only from 21% to 67% and from 29% to 81% primary interfaces, respectively, in six tested ISP topologies.

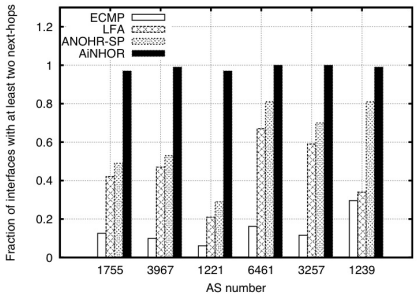


Fig. 6: Fraction of pI-D pairs with two routing options.

D. Path length distribution

AiNHOR has significantly improved the multipath capability that allows load balancing and increased fault-tolerance. However, adopting secondary interfaces for packet forwarding possibly leads to high path inflation which can increase traffic load over non-shortest paths. For that reason, we investigate the distribution of path length for different values of the maximum number of next-hops for each interface, denoted by K. Note that we have measured the *longest* path corresponding to K for each source-destination (S-D) pair. In other words, we show the *upper bound* of the path length for each S-D pair while packets might travel on much shorter paths in practice (where there is not the case that all primary links from the source node to the destination simultaneously fail).

Table III shows the distribution of path lengths in hop for K =1, 2 and 3. For $K = 1$, all paths from source nodes to destinations are shortest paths. Increasing K to 2 and 3 will allow more longer paths and therefore shift the average upper bounds up. Of all topologies, those upper bounds only increase from 7.5% (AS1239) to 40% (AS6461) for $K = 2$ and up to 100% (AS6461) for $K = 3$. In practice, those path lengths are still comparable to those of shortest path routing.

TABLE III: Path length distributions in hop

Allowed K	AS1755	AS3967	AS1221
1	5.2 ∓ 1.9	5.1 ∓ 1.9	4.8 ∓ 1.8
2	6.3 ∓ 2.6	6.9 ∓ 3.2	5.4 ∓ 1.9
3	6.8 ∓ 3.2	7.5 ∓ 3.9	5.7 ∓ 2.2
Allowed K	AS6461	AS3257	AS1239
1	4.3 ∓ 1.5	4.4 ∓ 1.6	4.4 ∓ 1.6
2	6.1 ∓ 2.6	6.1 ∓ 2.2	4.7 ∓ 1.8
3	9.1 ∓ 5.7	8.1 ∓ 4.3	5.2 ∓ 2.4

V. Related Work

Many recent solutions proposed in literature seek to increase the protection coverage with different approaches. We categorized those proposals into two main groups corresponding to two forwarding methods: the traditional IP forwarding and the interface-specific forwarding. Examples of the former are Tunnels [3], Not-via [4], O2 [18], PR [19], ANHOR-SP [9], MRC [5] and IDAGs [6]. The latter includes FIR [7], ESCAP [17], NISR [16] and LFIR [8].

Tunnels [3] and Not-via [4] suggest that IP packets should be encapsulated at the node adjacent to the failure. Those IP-in-IP packets then are tunneled to an unaffected node where they are forwarded to the destination. Promising 100% single failure coverage, the tunnel technique obviously requires high overheads in tunnel signalling.

O2 [18] and PR [19] are two routings designed for the centralized routing system. Both O2 and PR use the concept of "joker" links in which both directions of a link are used for mutual backups. O2 seeks to maximize the number of nodes with two next-hops while PR allows nodes to adopt *at least* two next-hops. In addition, PR is only suitable for small scale networks due to its high complexity. ANHOR-SP [9] introduces the concept of permutation routings and proposes a generic framework, from which various routing objectives for the traditional IP network can be realized. ANHOR-SP is designed for the distributed routing system due to its low complexity. ANHOR-SP, however, provides a coverage of single link faults which is significantly lower than AiNHOR.

MRC [5] and IDAGs [6] use multiple routing tables that cover all possible failures. Upon detecting a failure on the connected link, the affected node selects another routing table to avoid network interruption and marks the routing table index in its packets. Other node will examine such index in incoming packets and will select the same configuration.

FIR [7] first introduces the interface-specific forwarding concept and proposes the shortest path based algorithm to construct forwarding tables for incoming interfaces. FIR offers 100% single link fault coverage while a later version [20] provides full coverage of single node faults. ESCAPE [17] and NISR [16] share the same idea with FIR except that routing options are not necessarily bound to shortest paths. Those papers compute back-up ports for each affected router by solving an integer linear programming problem involving single component fault situations. Those methods, however, may produce forwarding loops when multiple failures occur [8].

VI. Conclusion

We have presented the permutation routing of interfaces as a method to increase robustness for IP networks with interface-specific forwarding. Our routing method is loop-free and does not introduce additional overheads for IP packets. In addition, it can work with the standard link state routing protocols such as OSPF or IS-IS due to its low complexity. Our method aims to approximate the optimality of the survivability by installing as many primary interface destination pairs with at least two next-hops as possible.

We have evaluated permutation routing of interfaces with simulations on six ISP topologies. The results show that permutation routings of interfaces offer protection coverage above 97% with realistic link weight settings.

References

[1] P. Francois, S. Bryant, B.Decraene, and M. Horneffer, "LFA applicability in SP networks," *RFC 6571*, June 2012.

[2] X. Y. M. Gjoka, V. Ram, "Evaluation of IP fast reroute proposals," *IEEE COMSWARE*, vol. 12, Jan. 2007.

[3] S. Bryant, C. Filsfils, S. Previdi, and M. Shand, "IP fast reroute using tunnels," *IEFT Internet Draft*, May 2008.

[4] M. Shand, S. Bryant, and S. Previdi, "IP fast reroute using not-via addresses," *Internet Draft (work in progress, expired in Dec. 2012)*.

[5] A. Kvalbein, A. F. Hansen, T. Čičic, S. Gjessing, and O. Lysne, "Multiple routing configurations for fast IP network recovery," *IEEE/ACM Transaction on Networking*, vol. 17, pp. 473–486, April 2009.

[6] S. Cho, T. Elhourani, and S. Ramasubramanian, "Independent directed acyclic graphs for resilient multipath routing," *IEEE/ACM Trans. Networking*, vol. 20, no. 1, pp. 153–162, Feb. 2012.

[7] S. Nelakuditi, S. Lee, Y. Yu, Z.-L. Zhang, and C.-N. Chuah, "Fast local rerouting for handling transient link failures," *IEEE/ACM Transactions on Networking*, vol. 15, pp. 359–372, April 2007.

[8] G. Enyedi and G. Rétvári, "A loop-free interface-based fast reroute technique," *Next Generation Internet Networks*, pp. 39–44, April 2008.

[9] H. Q. Vo, O. Lysne, and A. Kvalbein, "Permutation routing for increased robustness in IP networks," *Networking 2012*, vol. 1, pp. 217–231, 2012.

[10] H. J. Chao, "Next generation routers," *Proc. of the IEEE*, vol. 90, no. 9, pp. 1518–1558, 2002.

[11] B. K. Druken, "Line graphs and flow nets," *SDSU Theses and Dissertations*, November 2010.

[12] H. Q. Vo, O. Lysne, and A. Kvalbein, "Increased robustness with interface based permutation routing," Simula Networks and Distributed Systems Department, Tech. Rep. 2012-18. [Online]. Available: http://www.simula.no/publications/Simula.simula.1540

[13] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with rocketfuel," *SIGCOMM Comput. Commun. Rev.*, pp. 133–145, October 2002.

[14] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," *IEEE INFOCOM*, pp. 519–528, 2000.

[15] A. Nucci, N. T. S. Bhattacharyya, and C. Diot, "IGP link weight assignment for operational tier-1 backbones," *IEEE/ACM Transactions on Networking*, vol. 15, no. 4, pp. 789–802, August 2007.

[16] S. S. W. Lee, P.-K. Tseng, A. Chen, and C.-S. Wu, "Non-weighted interface specific routing for load-balanced fast local protection in IP networks," *IEEE ICC*, pp. 1–6, Jun. 2011.

[17] K. Xi and H. J. Chao, "IP fast rerouting for single-link/node failure recovery," *Broadnets*, pp. 142–151, Sept. 2007.

[18] G. Schollmeier, J. Charzinski, and A. Kirstadter, "Improving the resilience in IP networks," *High Performance Switching and Routing 2003 HPSR Workshop*, pp. 91–96, 2003.

[19] K.-W. Kwong, R. G. L. Gao, and Z.-L. Zhang, "On the feasibility and efficacy of protection routing in IP networks," *IEEE INFOCOM*, 2010.

[20] Z. Zhong, S. Nelakuditi, Y. Yu, S. Lee, J. Wang, and C.-N. Chuah, "Failure inferencing based fast rerouting for handling transient link and node failures," *IEEE Global Internet*, Mar. 2005.