Keep Forwarding: Towards K-link Failure Resilient Routing

Baohua Yang*†‡ Junda Liu§, Scott Shenker¶‖, Jun Li‡** and Kai Zheng*
* IBM China Research Laboratory, Beijing, China, 100193
† Dept. of Automation, Tsinghua University, Beijing, China, 100084
‡ Research Institute of Information Technology, Tsinghua University, Beijing, China, 100084
§ Google Inc., 1600 Amphitheatre Pkwy, Mountain View, CA 94043
¶ Dept. of EECS, University of California, Berkeley, CA 94704
‖ International Computer Sciences Institute (ICSI), Berkeley, CA 94704
** Tsinghua National Lab for Information Science and Technology, Beijing, China, 100084
{baohyang, zhengkai}@cn.ibm.com, junda@google.com, shenker@eecs.berkeley.edu, junl@tsinghua.edu.cn

***Abstract*—Handling link failures is the fundamental task of routing schemes. Routing protocols based on link state (*e.g.,* OSPF) require a global state advertisement and re-computation when link failure happens, and will cause inevitable delivery failures. To improve the routing resilience without introducing significant extra overhead, we propose a new routing approach, Keep Forwarding (KF) to achieve *k*-link failure resilience using inport-aware forwarding. KF is (i) flexible to handle multiple failures (or k-failure) with only small path stretch, (ii) efficient in recovery speed by instant and local lookup, (iii) bounded on memory requirement. Besides, the proposed approach is compatible with existing Internet protocols and routing infrastructures (*e.g.,* requires no packet labeling or state recording), and the pre-computation has a linear temporal complexity. Experimental results on real ISP and datacenter networks reveal that KF guarantees near-optimal resilience (99.9%∼100% for single failure and over 99.7% for multiple failures), with the average path stretch increment less than 5%.**

I. Introduction

With the rapid development of IP networking techniques, the Internet has already become the basic infrastructure for various information applications. More and more critical services are deployed in IP networks, requiring reliable packet delivery on top of the best-effort nature. Delivery disruptions, as short as several hundred milliseconds, may cause severe degradation in service quality. However, widely deployed routing schemes like OSPF [1] and IS-IS [2], both require global message exchanges and computation before packet delivery can be recovered from link failure. In addition, the re-computation time demanded is usually unacceptable [3]. More recently, centralized routing solutions [4]–[6] suggest that all routing computation is carried by a controller, who then pushes the results to affected routers. But there is an inevitable delay of at least the round trip time between the routers and the controller. Thus a *Local Failure Resilient* (LFR) routing method, which can provide fast failure recovery without the aid of neighboring routers or the controller, has more potential to meet the stringent reliability requirements.

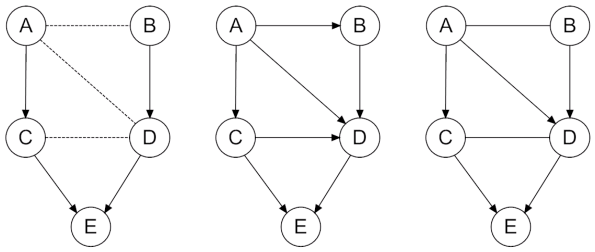
Existing LFR works fall into two categories by whether the packet labeling is needed. Schemes with packet labeling generally store extra bits into packets, *e.g.,* the forwarding history, and rewrite or remove them when necessary. Although they may provide good resilience, the change to data plane implies high cost for practical deployment. Label-free solutions, otherwise, only utilize information already available to routers, *e.g.,* destination IP and incoming port. They introduce minimal changes and cooperate with traditional IP routing. However, existing approaches can only handle single failure, while real networks may occur multiple failures simultaneously [7].

We argue that a practical LFR solution should address the following issues. i) *Resilience*: guarantee connectivity from single failure to multiple failures, while restricting the path stretch. ii) *Latency*: reduce the time interval before an alternative choice is deployed when a failure is detected. Ideally, it should be short enough so that no packet will be lost. iii) *Storage*: keep a moderate number of states in the router, so that existing router hardware can support effortlessly. This is especially important for production networks. iv) *Compatibility*: stay compatible with today's network protocols and existing infrastructure.

In this paper, we propose a novel labeling free LFR routing framework named as "Keep Forwarding". KF is designed to provide effective failure resilience for the general *k*-link failure case, and is built upon a new network model called *Partial Structural Network* (PSN). The advantages of the proposed approach and the contributions of this paper are summarized as follows.

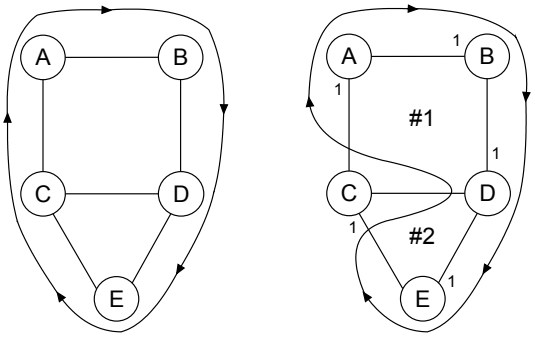
- *Flexible Resilience.* The proposed approach provides good resilience from single-failure to multi-failure cases with negligible path stretch.
- *Fast Recovery.* After failure happens, new forwarding choice is taken promptly from local table, which promises a line-speed recovery on existing router hardware.
- *Linear Scalability.* The temporal complexity of the pre-computation is proven linear, and the memory is also linearly bounded to the routing table.
- *Compatible Forwarding.* The proposed approach is well compatible with the traditional rule based routing

978-1-4799-3360-0/14/$31.00 ©2014 IEEE



(a) MST (b) DAG (c) PSN

Fig. 1. Comparison between MST, DAG and PSN, where "E" is the destination.



(a) KF traversal (b) c-KF traversal

Fig. 2. A simple example of KF traversal.

without any extra costs such as packet labeling or status recording.

- *Theoretical Contributions.* A new graph traversal (KF traversal) and a new network model (PSN) are proposed. Based on them, the *Imperfectness Theorem* is first proven on the k-failure resilience problem.

The remainder of this paper is organized as follows. Section II introduces the theoretical model for the k-failure resilience problem. Section III describes the routing design while the evaluation results are given in section IV. Section V summarizes the related efforts. At last, we conclude the paper.

II. Problem Formulation and Theory

This section first presents the PSN model and the KF traversal. After that, the k-failure resilient routing related theorems are presented. The detailed proofs of all theorems are given in the Appendix section.

A. Basic Assumption

For a given network W, the topology is modeled as a graph $G(V, E)$, where V, E means the set of nodes (or vertexes) and edges (or links) in G respectively. For every $e \in E$, e is bidirectional if there is no special indication.

B. Partial Structural Network

Several novel models have been proposed for network routing, such as the Minimum Spanning Tree (MST) and the Directed Acyclic Graph (DAG). For each destination, MST based schemes [1], [2] only utilize selected links (the ones on the spanning tree) and set them with determined directions. DAG based schemes [8], [9] utilize all links and set every one with a determined direction. DAG based schemes is proven to guarantee more routing resilience in our previous work [9]. Based on DAG, the Partial Structural Network (PSN) model is proposed, which utilizes all links but only sets determined directions to selected links.

Note that the PSN model is also per-destination based, thus with the same network, respective PSNs are generated for every destination. Fig.1 shows an example to compare PSN with MST and DAG. In the figure, "E" is the destination. In MST (Fig.1(a)), only the links on the shortest path are taken as the outgoing link. Hence each node has only one outgoing link. Once the outgoing link is down, the node will fail to deliver any packet. While in DAG (Fig.1(b)),some nodes ("A" and "C") have several outgoing links, thus a better resilience is achieved. For example, if "A-C" fails, "A" can easily pick up another outgoing edge (*e.g.,* "A-D") to recover the routing. However, there is still some deficiency to protect nodes with only one outgoing edge (*e.g.,* "B" and "D"). In the PSN model (Fig.1(c)), selected links (*e.g.,* "A-B" and "C-D") hold un-directed to achieve a better resilience. For example, when "D-E" fails, "D-C" is still available as outgoing for "D". In summary, PSN implies higher flexibility in failure protection.

The primitive idea of utilizing the links as un-directed is not new, however it still remains two open problems: "*Which links should be selected as un-directed*?" and "*How to determine the directions of the directed links*?" The construction algorithm of PSN will be detailed in Section III-B1.

C. KF Traversal

1) Definitions and Theorems:

***Definition** 1 (KF Traversal):* With an undirected graph $G(V, E)$, for every $v \in V$ and $e \in E$ (e can be utilized in both directions), KF traversal visits v at least once, and visits e at most once in each direction. [1]

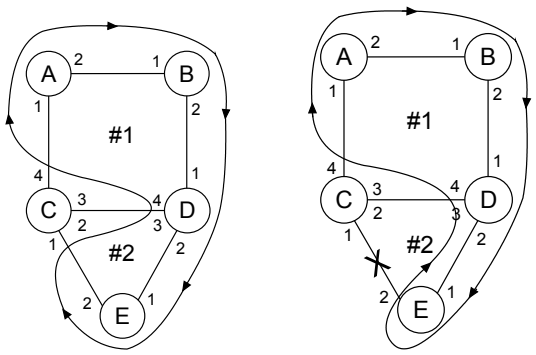
There may exist more than one KF traversal for a graph. Fig.2 shows an example, where two different KF traversals exist on a 5-node graph. The traversals in Fig.2(a) and Fig.2(b) generates node visiting sequences of "A-B-D-E-C-A" and "A-B-D-E-C-D-C-A" respectively. Such visiting circuits generated by KF traversal as KF circuits.

Another question is whether the KF traversal always exists on arbitrary graphs. A sufficient condition is given as follows.

***Theorem** 2.1 (KF traversal Existence):* For any undirected graph $G(V, E)$, if G is connected, the KF traversal must exist.

Theorem (2.1) reveals the general existence of the KF traversal. Among all the KF traversals on a graph, the ones visiting every e, are defined as "complete KF traversals", or *c-KF traversals*.

[1] We will compare the definition with Euler traversal and Hamiltonian traversal later.



(a) c-KF traversal (b) c-KF traversal with failure

Fig. 3. c-KF shows some potential in failure resilience.

TABLE I. FORWARDING ENTRIES IN NODE "D"

Incoming Edge	1st	2nd	3rd
B-D	D-E	D-C	D-B
E-D	D-C	D-B	D-E
C-D	D-B	D-E	D-C

***Definition** 2 (c-KF traversal):* With an undirected graph $G(V, E)$, a complete KF (c-KF) traversal is the KF traversal that visits every $e \in E$.

Take Fig.2 for example, Fig.2(b) shows a c-KF traversal, while Fig.2(a) does not. Similarly, the existence of c-KF traversal is described as follows.

***Theorem** 2.2 (c-KF Traversal Existence):* For any undirected graph $G(V, E)$, if G is connected, there must be at least one c-KF traversal.

2) Failure Resilience: Theorem (2.2) reveals c-KF traversal's existence. A further question is how to find one for a given graph. The solution is designed following a greedy principle: traverse every path or circle successively based on Depth First Search. Take Fig.2(b) for example, there are two non-inclusive circles: #1 ("A→B→D→C→A") and #2 ("D→E→C→D"). Suppose the traversal starts from an arbitrary node *e.g.,* "A", in clockwise. When the traversal reaches "D" following "A→B→D", a new circle (#2) is met, and then traverse circle #2 first. After the traversal on #2 ("D→E→C→D") finished, continue the traversal on circle #1 with "D→C→A". At last, we obtain the c-KF circuit as "A→B→D→E→C→D→C→A".

The generated c-KF traversal shows a good potential in failure-resilience. In Fig.3(a), each port can obtain an id based on the visiting (enter or leave) order. Take "D" for example, the traversal visits it orderly from edge "B-D" and"D-E" once, and "C-D" twice. Then ports from "B-D" and "D-E" are set with id 1 and 2 respectively; and port from edge "C-D" is with id 3 and 4. Hence each node gets an ordered sequence of its ports, *e.g.,* "D-B, D-E, D-C" for "D". By rotating these ports, a forwarding table is constructed, as shown in Table (I).

The first column denotes the incoming port, while following columns indicate the forwarding candidates with priority order. Take "B-D" as the incoming link, "D-E", "D-C" and "D-B" are the corresponding outgoing links in descending order. A c-KF traversal generates such a forwarding table for each node. As shown in Fig.3(b), when the outgoing link, *e.g.,* "E-C" at "E" is broken, "E" will pick the next available port, *i.e.,* "E-D",.

TABLE II. TRAVERSAL COMPARISON*

Traversal	Node	Edge	Each direction	Existence
Hamilton	= 1	0, 1	0, 1	Topo dependent
Euler	≥1	1	0, 1 respectively	Topo dependent
KF	≥1	0, 1, 2	0, 1	Yes
complete KF	≥1	1, 2	0, 1	Yes

* All traversals are on undirected connected graphs.

As a summary, Table II compares KF and c-KF traversal with the well-known Hamilton traversal [10] and Euler traversal [11]. In Table II, the first and the second column mean the visiting times on each node and edge (on both directions), while the third column means the visiting times on each edge from one direction. The last column tells the existence. Compared with both Hamilton traversal and Euler traversal, KF traversal implies more flexibility in existence by topology independence.

D. Reachability

There are several ways to measure the resilience of a routing scheme, *e.g.,* the protected node ratio for single failure [12]. This paper employs reachability for comprehensive evaluation. With D, F_k denoting the destinations and the entire failure space respectively, the reachability R is given in Equation (1).

$$R_{W,k} = \sum_{F_k} \sum_{D} N_{delivered} \Big/ \sum_{F_k} \sum_{D} N_{connected} \tag{1}$$

Where $N_{delivered}$ means the number of other routers in network W (excluding the destination itself) that successfully deliver packets to the destination, and $N_{connected}$ is the number of other routers that are connected with the destination. It is obvious that $0 \leq R \leq 1.0$, and a higher R means a better failure resilience. When W becomes disconnected with failures, no routing framework can protect the reachability. Hence, we focus on the non-disconnected failure cases.

E. K-failure Resilient Routing Problem

The goal of the k-failure resilient routing problem is to maximize R over all failure cases.

***Definition** 3 ("Perfect" Resilience):* For an arbitrary network topology $G(V, E)$, if a routing scheme can always protect $R = 1.0$ against every k-failure case, then the routing provides "perfect" resilience.

If a "perfect" resilient routing exists, the ideal 100% resilience will be achieved. However, the following theorem implies that the "perfect" resilience cannot always be guaranteed by static rule based routing solutions[2].

***Theorem** 2.3 (Routing Imperfectness Theorem):* For an arbitrary network topology $G(V, E)$, if multiple failures may happen, then there is no static rule based routing that always

[2]It means the forwarding decision is only based on the precomputed rules stored in the local routing tables which will not change, and there is no extra control information.

guarantees "perfect" resilience, even when G keeps connected after failures.

Theorem (2.3) unveils that for some W, there may exist a non-disconnecting failure case which cannot be protected, *e.g.,* transient loop may exist with multiple failures. These topologies are named as "imperfect" topologies.

It is natural to ask what type of topology is "imperfect"? From Theorem (2.3), we can deduce Theorem (2.4) to answer the question (see Appendix B).

***Theorem** 2.4 (Graph Imperfectness Theorem):* For an arbitrary graph $G(V, E)$, if any component of G is "imperfect", there will be no static rule based routing guaranteeing the "perfect" resilience, even G keeps connected after failures.

Theorem (2.4) proves that with the simple local rule based routing, there is no way to guarantee the perfect resilience on every nodes. In this paper, in order to provide near-optimal performance while guaranteeing the simplicity, KF routing is proposed by leveraging the PSN model's resilience potential.

III. Routing Design

In this section, we first propose the employment of inport-aware routing in KF, and then describe the pre-computation of routing table and the lookup procedure. After that, the precomputation complexity is analyzed. In the end, we discuss several relevant problems.

A. Inport-Aware Routing

The basic procedure of the traditional IP routing can be formulated as: "*Destination→Outport(s)*"[3]. While with inport-aware routing, both the destination IP and the ingress port are employed for the routing lookup. Besides, the routing action consists of a sequence of egress ports with descending priority. Similarly, the procedure can be formulated as "*Destination+Inport→ Ordered Outport Sequence*".

Inport-aware routing is quite practical with existing router architecture, as most existing routers maintain a routing table at each line card of the interface for lookup efficiency [13], [14]. The only difference is that for traditional routers, the same tables are duplicated in different line cards, while the inport-aware routing tables may vary in different line cards. Thus the actual memory required by inport-ware routing is d times of that by traditional routing (where d is the router's degree). Suppose the states number of traditional IP routing is $O(N)$ (N is the number of routers), for inport-aware routing it will be $O(d^2 \cdot N)$. On the other hand, the lookup complexity is not increased, as only the corresponding routing table will be searched for the incoming packet.

B. Routing Pre-computation

The pre-computation phase for an inport-ware routing includes three steps. First, a PSN is built for each destination. After that, for each PSN, every link is set with a priority. At last, the routing table is generated based on the priority.

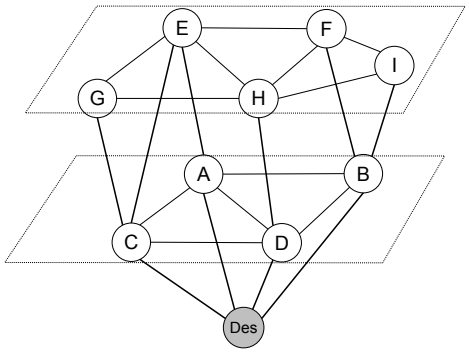


Fig. 4. Example of building PSN.

1) PSN Building: The building of PSN is with each destination. For each destination, the building procedure is illustrated in Algorithm 1.

Algorithm 1 PSN Building

Input:
network topology $G(V, E)$
destination node d
empty Alayers set S
Output:
PSN p

```
/*Generate weight for every node.*/
for e in G do
    w = getWeight(e, d)
    S.add(e, w)
end for
/*Set type for every link.*/
for α in S do
    p.updateMlinks(α)
    p.updateAlinks(α)
end for
```

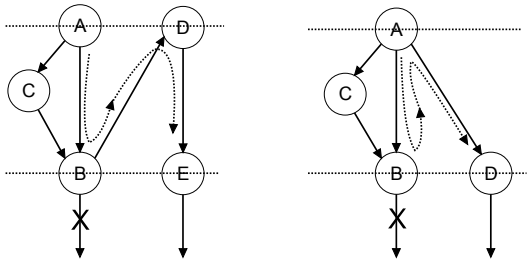
In Algorithm 1, the node weight is calculated based on the distance (*e.g.,* the hop counts) to the destination. Nodes with the same weight are grouped into an *A-layer* (Aid layer). Furthermore, links within the A-layer are named as *A-links*, while links between two A-layers are *M-links*. M-links consist of two types by the direction: *Down-links* that from higher A-layer to lower one, and *Up-links* conversely.

Fig.4 shows an example, where "Des" is the destination. "A, B, C, D" and "E, F, G, H, I" belong to A-layer #1 and #2 respectively. "A-B, A-C, A-D, B-D, C-D, E-F, E-G, E-H, F-I, F-H, G-H, H-I" are A-links while "A-Des, B-Des, C-Des, D-Des, E-A, E-C, F-B, G-C, H-D, I-B" are M-links. Each M-link is set with two directions. Take link "A-Des" for example, "A→Des" is a Down-link, while "Des→A" is an Up-link.

With PSN, basic per-destination routing can be achieved by forwarding along existing M-links. When there is no M-link, the delivery is protected by the A-links.

2) Priority Calculation: For routers with several links of the same type (*e.g.,* A-link), a priority is needed to generate

[3]In the case of the Equal Cost Multiple Path (ECMP), more than one outport can be assigned to one destination, however, the multiple path routing is not involved in this paper.



(a) Smart selection of Up-link.

(b) Smart selection of Down-link.

Fig. 5. Example of the Smart Selection Algorithms.

the routing table. An intuitive way is to calculate the potential to reach the destination. Suppose the link quality is the same between each other, it is clear that a link towards a high-potential neighbor router should have a higher priority; hence it turns out to calculate the routers' priority. Here we propose a basic method by the number of outgoing links. It is intuitive that a router with more outgoing links should have higher potential to reach the destination. Thus, the router's priority can be calculated with the weight summation of all the outgoing links. Since outgoing links may vary from M-links and A-links, different weights need to be set with different link types.

Note the priority can be calculated by other parameters, *e.g.,* disjoint paths, link bandwidth, to meet various requirements.

3) Route Table Generation: The route table is generated for per-destination and per-inport. Since a port may connect to router of a higher, lower or the same A-layer, the inport is classified as Up-inport, Down-inport or A-inport respectively. The generation will carry out according to the inport type.

Up-inport: If there are available Down-links, pick up the one with the highest priority. If not, check available A-links. If neither Down-link nor A-link is available, perform the *smart Up-link selection* algorithm (see Section III-B4). Otherwise, forward back through the ingress port.

Down-inport: Select a Down-link following the *smart Down-link selection* algorithm. If no available Down-link exists, pick the available A-link with the highest priority. Otherwise, check the Up-links. At last, forward back through the ingress port.

A-inport: Pick an available Down-link with the highest priority. If failed, select an A-link by the *smart A-link selection* algorithm. If failed again, check the Up-links to pick available one with the highest priority. Otherwise, forward back.

With these three basic principles, forwarding candidates for each inport are generated.

4) Smart Selection Algorithms: Before describing the smart selection algorithms, we first define *key* router, *key* link and *plain* router, *plain* link.

***Definition** 4 (key router, link and plain router, link):* A router is a *key* router if it connects more than two links or it is the destination; otherwise it is a *plain* router. The link connected to a *key* router is a *key* link; otherwise a *plain* link.

Plain routers and links can act as relays, thus they can forward as a straight chain. Along a key link or a plain chain, a key router will be reached finally, which is the true sink of the forwarding. Links are grouped by their true sink. Based on these definitions, the smart selection procedure is described in Algorithm 2.

Algorithm 2 Smart Selection

Input:
ingress link in
Up link set U
Down link set D
Output:
selected link l

```
1: /*Smart Up-link selection.*/
2: if in.type == Up-link then
3:    for l in U && not l.sinkGid == in.sinkGid do
4:       if not l.isFailed then
5:          return l
6:       end if
7:    end for
8:    for l in U && l.isKeylink do
9:       if not l.isFailed then
10:         return l
11:      end if
12:   end for
13:   /*Smart Down-link selection.*/
14: else if in.type == Down-link then
15:   for l in D && not l.sinkGid == in.sinkGid do
16:      if not l.isFailed then
17:         return l
18:      end if
19:   end for
20:   /*Smart A-link selection.*/
21: else
22:   K = getKFTraversal(in.ALayer)
23:   l = nextLink(K, in)
24:   return l
25: end if
```

Fig.5(a) shows an example of the smart Up-link selection algorithm. "B" has no available Down-links or A-links. Suppose packets come from Up-link "A-B", "B" tries to forward packets via "B-D", which belongs to a different group from "A-B". On the other hand, if "B-D" is also failed, "B" only has two Up-links in the same group, then "B" will forward packets via a key link, or "A-B".

Fig.5(b) shows an example of the smart Down-link selection algorithm. The Down-link of "B" is failed. When "A" forwards packets to "B", "B" forwards back via "B-A". "A" will receives the packet from "B-A", "A" then knows that "B" has no available path to reach the destination, thus "A" will pick a Down-link from a different group, *e.g.,* "A-D".

C. Routing Lookup

With the generated routing table, the lookup procedure is similar as the traditional routing lookup on every inport. Suppose a packet comes from inport *in*, and the routing table on *in* is T_{in}. The *Longest Prefix Match* is performed to find the best-matched rule. After that, the first available outport will be picked from the action field. When some link fails, the router

will remove the corresponding port from its action field. The pseudo code is given in Algorithm 3.

Algorithm 3 Routing Lookup

Input:
arrived packet p
ingress port in
Output:
output port out

```
1: des = getDes(p) /*Get the destination.*/
2: rTable = getRouteTable(in)
3: rule = rTable.doLPM(des) /*Longest Prefix Match.*/
4: for egress in rule.action do
5:    if not egress.isFailed then
6:       out = egress
7:       return out
8:    end if
9: end for
```

D. Precomputation Complexity

In the precomputation procedure (build PSN and generate route table), every node will be taken as the destination once. For each destination, every link in the network will be labeled with a type within constant operations. The theoretical complexity should be $O(|V| \cdot |E|)$, where $|V| \cdot |E|$ means the network size. This size-linear complexity provides a good scalability. Evaluation results in Section IV-C supports the inference.

E. Discussion

1) Loop Probability: One question is the probability of local loop. As Theorem 2.4 indicated (see Section II-E), no static rule based routing can protect resilience while guaranteeing no loop. Although loop may happen scarcely in special failure case, we argue that the loop will never happen in normal case. In KF routing, as long as one M-link is available for an A-Layer, the traffic will be forwarded to the lower A-layer finally. In practise, the transient local loop is rarely generated by KF (Experimental results in IV also prove this.). Moreover, the routers can insert specific rules in advance to avoid possible loops, or take advantages of schemes such as Time-To-Live.

2) Network Disconnection: Another question is the network topology disconnection. When the network itself is disconnected in topology, it is clear that no scheme can protect the routing resilience. KF utilizes the Time-To-Live mechanism to prevent exhausting the bandwidth by endless trying. On the other hand, monitoring schemes can be taken to help handle the issue. In graph theory there is a theorem that every $2k$-link-connected graph has k link-disjoint spanning trees [15], thus k-link failure can be perfectly handled on $2k$-link-connected graphs. However, the proposed approach is designed not limited to specific graphs, *e.g.,* dense graphs, but also for general network topologies.

3) Weighted Links: Existing schemes such as OSPF/IS-IS supports links assigned with weight (*e.g.,* capacity). Basic version of KF is designed based on hop counts due to several reasons. First, compared with the recovery time (usually longer than minute), the forwarding latency (usually milliseconds) is much less by several orders. Thus, we would like to overcome the recovery time with low overhead. Besides, production datacenter networks tends to employ techniques supporting flat and any-to-any connectivity [16], [17], to guarantee high performance with simple configuration. In this case, all edge switches and paths are considered as equal with each other.

TABLE III. Topologies Used for Evaluation

Name	Topology	Nodes	Edges	Avg. Degree
AS1	AS1221	83	131	3.16
AS2	AS1239	361	1479	8.19
AS3	AS1755	111	234	4.22
AS4	AS3257	151	288	3.81
AS5	AS3967	91	180	3.96
AS6	AS7018	382	1299	6.80
DC1	Cisco	76	160	4.21
DC2	FatTree	80	256	6.40
DC3	VL2	88	256	5.82

4) Traffic Optimization: Although KF is designed not specifically for traffic optimization goal, it still protects good network performance. First, KF always tries to deliver the traffic along the shortest path as much as possible when failures happen, which guarantees a shortest delivery latency. Besides, KF supports to pickup backup paths with distributing traffic. By examining the traffic load of each link, priorities can be calculated in order to achieve better network performance. Hence, KF is applicable in conjunction with traffic engineering.

IV. Evaluation and Analysis

A. Evaluation Setup

In order to collect reliable results, real-life topologies are utilized. The inherent characteristics of every topology are shown in Table III. Among the topologies, the AS series are from ISP backbone networks [18], while the DC ones are recommended for enterprise and datacenter networks [19].

With these real network topologies, the evaluation is carried out under k-failure ($k = 1, 2, 3$) cases[4]. The experimental platform is an x86 based PC (2.20 GHz CPU with 2.5 GB memory) running Linux 2.6. All the code is written in Python, allowing quick development at the cost of slower running speed. We have also tried several widespread evaluation tools, *e.g.,* NS2 and Mininet. However, the k-failure problem is a specific problem, requiring fast and large-scale calculation, thus general platforms are not suitable.

B. Reachability Results

Fig.6 shows the average reachability result (the Y-axis starts from 99%). It shows that on all topologies, KF always keeps a high reachability very close to 100%. For $k = 1$ (single failure), all datacenter topologies are perfectly protected, *i.e.,* achieve 100% reachability. Among the ISP topologies, AS1 is protected with 100% reachability and other topologies are protected with over 99.8% (mostly over 99.9%) reachability.

When k increases, the reachability remains near perfect. All datacenter topologies have over 99.99% reachability even with

[4]Since the number of evaluation cases is explosively increased with k, now we are able to collect results with $k = 1, 2, 3$ under the time limitation. We have started to examine $k > 3$ cases, and will be able to present more results soon.

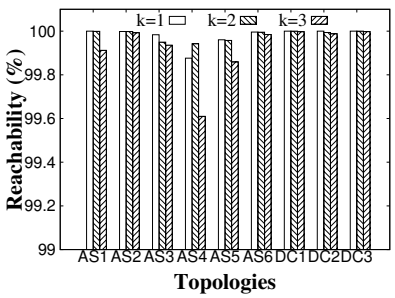


Fig. 6. Reachability results with k-failure case.

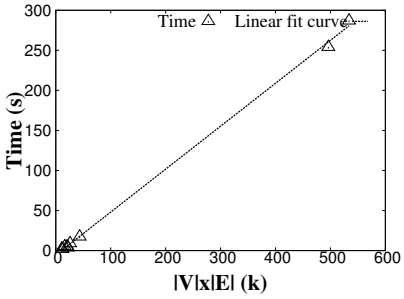


Fig. 7. Results of the pre-computation time.

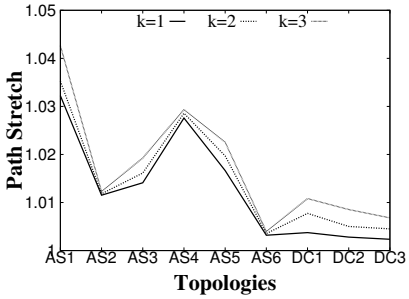


Fig. 8. The average path stretch of KF routing.

$k = 3$ concurrent failures. The ISP results show two interesting features: first, the reachability stays close to 100% even with multiple failures. More importantly, the results are stable from single failure to multiple failures. This implies that KF is able to handle more complicated failure cases.

C. Pre-computation Time

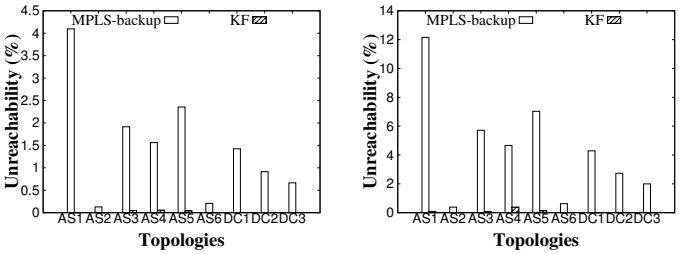
Fig.7 illustrates the precomputation time on all topologies. The precomputation time is less than 20s for most topologies (AS1, AS3, AS4, AS5 and all datacenter ones). On very huge topologies (AS2, AS6 are so large that no previous work can run on them within a reasonable time), the pre-computation time is still acceptable. In the evaluation, all the computation is implemented without any parallel processing. However, as the precomputation of different destinations is naturally isolated from each other, the performance can be simply accelerated using parallel processing. On average, the per-destination calculation is less than 0.1s. Hence with parallel techniques, the pre-computation can be completed nearly instantly.

Fig.7 also demonstrates a linear fit curve over the $|E| \times |V|$ on all topologies. The adjusted fit coefficients $R^2 = 0.999$. We notice that there are two large size topologies staying far from the other smaller ones. The curve only on the small topologies is also fitted, where the adjusted fit coefficients R^2 is nearly 0.9. These results strongly support that the temporal complexity of KF's pre-computation is linear to the $|E| \times |V|$.

D. Path Stretch

The average path length is compared between KF (with failures) and the shortest path routing (without failure). Note without failures, KF achieves the same path length with the shortest path routing (see Section III-E4). Path stretch is adopted for a clear illustration, which is defined as the path length ratio. Fig.8 shows the results. Under all k-failure cases among all topologies, KF only introduces insignificant increase (less than 5%) on the average path length. All these results demonstrated that KF routing achieves a near optimal forwarding path length. Besides, with k increasing, the path stretch is also increased. This is because with more failures happened KF will explore more A-links to recovery from the failure, which will increase the path length.

E. Compare Results



(a) k=2 failures happen. (b) k=3 failures happen.

Fig. 9. Comparison with MPLS-backup on un-reachability.

1) Compare with MPLS: KF is compared with the de-facto routing protocol, Multi-Protocol Label Switching (MPLS) with backup paths, *i.e.,* MPLS-backup, where a backup path is stored for each single-link failure case[5]. When no failure happens, the packets will be forwarded along the shortest path. When a failure happens, MPLS-backup will switch to the corresponding backup path for delivery. In order to illustrate clearly, the un-reachability (defined as $100\% - reachability$) is taken as the comparison parameter. The less the un-reachability is, the better the resilience will be. Fig.9 shows the comparison results. From Fig.9, MPLS-backup (the blank columns) gains at least two orders of magnitude un-reachability as KF (the decorative columns). Take AS1 for example, with two failures, 4.10% of traffic is failed to reach the destination by MPLS-backup, while for KF, it is less than 0.001%.

Another observation is that MPLS-back achieves better results on AS2, AS6 (whose average degrees are large) than on other topologies,while on the least-average-degree topology (AS1), the result is also the worst. This implies MPLS-backup is degree-sensitive. KF implies a much better stability.

2) Compare with SNH: In order to compare with the best recent work SNH [12] that we know, we also show the protected node ratio across all destinations. Notice that the evaluations in SNH are taken 3 reduced AS topologies: AS1, AS3 and AS5, with the average degree 2.90, 3.70 and 3.72 respectively ([12] does not provide the reduction method).

As defined in [12], a node is said to be protected with a destination if packets from it can reach the destination under

[5]Although the multiple failure protection is possible on special topologies, it requires quite complicated computation and huge storage for MPLS, which is unacceptable for today's routers

every single failure case. On the reduced AS1, SNH provides less than 50% node ratio, while KF provides 100% protection on the original topology. On the reduced AS3 and AS5, SNH's results are around 84% and 90% respectively, while KF guarantees a high ratio of 96.3% and 93.6% respectively. On other large topologies (AS2, AS4 and AS6) that SNH does not give the evaluation results, KF also provides a good protection of over 90%. Although SNH is not interface specific, KF only needs an additional information, the inport. On the other hand, SNH is NP-hard in pre-computation, which results in more time cost. For example, SNH requires 1.82 hour on AS1 with a Pentium Xeon 2.66 GHz machine [12].

V. Related Work

Improving IP network resilience has drawn great attention from both industry and academia. Various approaches have been proposed to improve the resilience of Internet routing, including the IPFRR related mechanisms [20]–[24] and multi-path or multi-homed routing [25]. Due to space limitation, we summarize the most relevant efforts as follows.

Recent works without packet labeling include [13] and [12]. In [13], an interface-specific forwarding scheme called FIR is proposed, which utilizes the incoming port for backup path computation. When only single link failure notification is suppressed, a loop-free path is generated to forward packets to its destination if such a path exists. However, this design only works on the single failure case and requires a complicated pre-computation. SNH [12] utilizes a secondary next-hop to provide a transient backup path after failure while waiting for the update from the centralized server . Though for single failure it provides better resilience than previous methods [8], [26], the results may not be good enough on real network topologies. Besides, the multi-failure case is not discussed.

Works in [27] and [28] utilize packet labeling to be able to handle multiple failures. The Failure-Carrying Packets (FCP) [27] has been proposed to allow routing recovery with multiple failures, however, this technique also requires considerable overhead in the packet headers, as well as the extra detection and computation cost on each router, to obtain the new path when receiving a failure carrying packet. [28] presents a Packet Re-cycling technique that employs extra bits in the packet header to reroute for non-disconnecting failures. This work is a novel rerouting approach to handle failures on an orientable topology network. However, it brings the cost of extra bits and packet modifications. With the diameter of the network is d, [28] requires $log_2(d)$ extra bits in the packet header.

Table IV summarizes KF and the previous works.

VI. Conclusion

In this paper, we for the first time present a formal study of the general k-failure resilient routing problem. A novel and practical k-failure resilient routing framework, called "Keep Forwarding", is proposed, which only utilizes local static rules to achieve fast recovery when failures happen. KF is compatible with the existing IP networks without demanding packet labeling or extra control message. Compared with the shortest path routing, KF's path length only increases slightly under failure. Besides, the storage requirement is linearly bounded with existing routing, and the temporal complexity of pre-computation is linear with the network size. Experimental results on real ISP and datacenter networks prove that KF provides high reachability and good delivery latency with multiple failures. Furthermore, a new network model and several graph theorems are proposed to support the approach.

References

[1] J. Moy, "OSPF version 2," *RFC 2328*, April 1998.

[2] D. Oran, "OSI IS-IS intra-domain routing protocol," *RFC 1142*, February 1990.

[3] P. Francois, C. Filsfils, J. Evans, and O. Bonaventure, "Achieving sub-second IGP convergence in large IP networks," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 3, pp. 35–44, 2005.

[4] A. Greenberg, G. Hjalmtysson, D. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, "A clean slate 4D approach to network control and management," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 5, pp. 41–54, 2005.

[5] H. Yan, D. Maltz, T. Ng, H. Gogineni, H. Zhang, and Z. Cai, "Tesseract: a 4D network control plane," in *Proceedings of the 4th USENIX conference on Networked Systems Design and Implementation*, 2007, pp. 369–382.

[6] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[7] L. Shen, X. Yang, and B. Ramamurthy, "Shared risk link group (SRLG)-diverse path provisioning under hybrid service level agreements in wavelength-routed optical mesh networks," *IEEE/ACM Transactions on Networking (ToN)*, vol. 13, no. 4, pp. 918–931, 2005.

[8] S. Ray, R. Guérin, K. Kwong, and R. Sofia, "Always acyclic distributed path computation," *IEEE/ACM Transactions on Networking (ToN)*, vol. 18, no. 1, pp. 307–319, 2010.

[9] J. Liu, B. Yang, S. Shenker, and M. Schapira, "Data-driven network connectivity," in *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, 2011.

[10] William Rowan Hamilton, "Account of the Icosian Calculus," *Proceedings of the Royal Irish Academy*, vol. 6, 1858.

[11] L. Euler, "Solutio problematis ad geometriam situs pertinentis," *Commentarii Academiae Scientiarum Imperialis Petropolitanae*, vol. 8, pp. 128–140, 1736.

[12] K. Kwong, L. Gao, R. Guérin, and Z. Zhang, "On the feasibility and efficacy of protection routing in IP networks," *IEEE/ACM Transactions on Networking (ToN)*, vol. 19, pp. 1543–1556, October 2011.

[13] S. Nelakuditi, S. Lee, Y. Yu, Z. Zhang, and C. Chuah, "Fast local rerouting for handling transient link failures," *IEEE/ACM Transactions on Networking (ToN)*, vol. 15, no. 2, pp. 359–372, 2007.

[14] Cisco. Virtual routing and forwarding. [Online]. Available: http://www.cisco.com/en/US/docs/net_mgmt/active_network_abstraction/3.7/reference/guide/vrf.html

[15] W. Tutte, "A theory of 3-connected graphs," *Indag. Math*, vol. 23, pp. 441–455, 1961.

[16] QFabric. Juniper Networks. [Online]. Available: http://www.juniper.net/us/en/dm/datacenter/

[17] FabricPath. Cisco Networks. [Online]. Available: http://www.cisco.com/en/US/prod/collateral/switches/ps9441/ps9402/white_paper_c11-605488.html

[18] Rocketfuel Project. [Online]. Available: http://www.cs.washington.edu/research/networking/rocketfuel/

[19] A. Greenberg, J. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, P. Patel, and S. Sengupta, "VL2: A Scalable and Flexible Data Center Network," in *Proceedings of the ACM SIGCOMM*, 2009, pp. 51–62.

[20] A. Li, P. Francois, and X. Yang, "On improving the efficiency and manageability of NotVia," in *Proceedings of the ACM CoNEXT conference*, 2007.

TABLE IV. COMPARISON WITH PREVIOUS WORKS

Approach	Single-failure	Multi-failure	Labeling Free	Compatibility*	Computation Complexity	Interface-specific
IPFRR	Yes	No	Mostly	Mostly	Linear (Mostly)	No (Mostly)
FIR	Yes	No	Yes	Yes	$O(\|E\| \cdot \|V\| \cdot log^2\|V\|)$	Yes
SNH	Yes	No	Yes	Yes	NP-hard	No
FCP	Yes	Yes	No	No	N/A	No
Re-cycling	Yes	Yes	No	No	Generally NP-hard	No
KF	Yes	Yes	Yes	Yes	Linear	Yes

* Whether the scheme requires heavy modifications to the existing router architecture or not.

[21] A. Atlas and A. Zinin, "Basic specification for IP fast-reroute: Loop-free Alternates," *RFC 5286*, September 2008.

[22] S. Kini, S. Ramasubramanian, A. Kvalbein, and A. Hansen, "Fast recovery from dual link failures in IP networks," in *Proceedings of the IEEE INFOCOM*, 2009, pp. 1368–1376.

[23] A. Li, X. Yang, and D. Wetherall, "SafeGuard: Safe forwarding during route changes," in *Proceedings of the ACM CoNEXT conference*. ACM, 2009, pp. 301–312.

[24] M. Shand and S. Bryant, "IP fast reroute framework," *RFC 5714*, January 2010.

[25] M. Motiwala, M. Elmore, N. Feamster, and S. Vempala, "Path splicing," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 27–38, 2008.

[26] C. Reichert and Y. Glickmann, "Two Routing Algorithms for Failure Protection in IP Networks," in *Proceedings of the 10th IEEE Symposium on Computers and Communications*, 2005, pp. 97–102.

[27] K. Lakshminarayanan, M. Caesar, M. Rangan, T. Anderson, S. Shenker, and I. Stoica, "Achieving convergence-free routing using failure-carrying packets," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 241–252, 2007.

[28] S. Lor, R. Landa, and M. Rio, "Packet re-cycling: Eliminating packet losses due to network failures," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010.

APPENDIX A
PROOF OF THEOREM (2.1) AND THEOREM (2.2)

Let $G(V, E)$ be an arbitrary connected graph. Suppose C_{el}, C_{kf}, C_{ckf} means the Euler circuit, KF circuit and complete KF circuit respectively. If $\forall v \in V$, $degree(v)$ is even, then G has a C_{el} [9]. Let $C_{ckf} = C_{el}$. Proof is done.

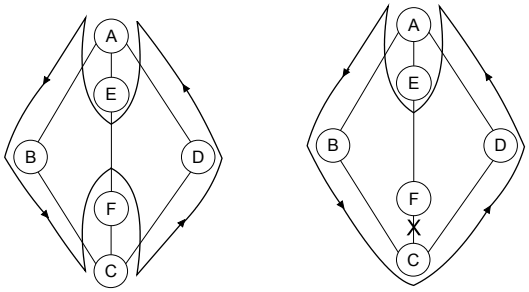
If for some $v \in V$, $degree(v)$ is not even, combine these v as a set $V_{(odd)}$, then $|V_{(odd)}|$ must be even, because $\sum_{v \in V} degree(v) = 2|E|$ is even. Hence $v \in V_{(odd)}$ can be combined into pairs.

Here we have a lemma.

Lemma A.1: For $V_{(odd)}$, there must be a pair combination, by which for a pair i, an odd path $p_i (i = 1, 2 \ldots, \frac{|V_{(odd)}|}{2})$ can be built, so that p_i and p_j have no overlap e if $i \neq j$, where an odd path means only its two ends are odd nodes.

Lemma (A.1) can be proven recursively. Consider $G(V, E)$, where $|V_{(odd)}| \neq 0$, first pick an odd path,*e.g.,* p_i, then remove every e for $e \in p_i$. After that, several connected subgraphs S_G may be generated. If for every $G_i(V_i, E_i) \in S_G$, $|V_{i(odd)}| = 0$, then the proof is done. Otherwise, suppose there is a $G'(V', E') \in S_G$, where $|V'_{(odd)}| \neq 0$. Repeat the removing procedure on $G'(V', E')$ until for every $G_i(V_i, E_i) \in S_G$, $|V_{i(odd)}| = 0$. Because every removed e is only processed once, there's no e belonging to p_i and p_j when $i \neq j$.

With Lemma (A.1), doubling edges on each p_i, a new graph G' is generated,which is even. Then G' must have a C_{el}, which is also a C_{ckf} on G. Proof is done.



(a) Normal Forwarding (b) Failure Forwarding

Fig. 10. Example for Theorem (2.3)'s Proof.

APPENDIX B
PROOF OF THEOREM (2.3)

Before giving the proof, a lemma is proposed as follows.

Lemma B.1: Static rule based traversal cannot always guarantee an overall node visiting against all failure cases with arbitrary topology, where overall node visiting means to visit every node at least once.

To prove this lemma, Fig.10 shows an example. To visit every node, the traversal should visit the path p ="A-E-F-C" from "A" or "C", thus there must be one side from which m nodes on p are visited, where $m \geq 2$, as there are 4 nodes on p. Suppose the traversal from "C" visit 2 nodes as "C-F", the entire traversal is shown in Fig.10(a). Then fail "F-C", "F" will be never visited, as shown in Fig.10(b). Similarly, if we visit $m \geq 2$ nodes on p from "A", "E" will be not visitable by failing edge "A-E". In general, if there is a "circle-circle" structure in the topology, the traversal is not always guaranteed for every failures case. Such a topology is "imperfect".

With Lemma (B.1), an example can be constructed to prove Theorem (2.3). Consider a connected graph $G(V, E)$, where $|V| = N + 1$. N nodes of G combine as an A-layer L with circle-circle structure, and the left node d is the destination. $\forall v_i \in L (i = 1 \ldots N)$, let $e(v_i, d)$ connects v_i and d. Suppose we fail $N - 1$ arbitrary edges, *e.g.,* $e(v_i, d)$ for $1 \leq i \leq N - 1$. Note G keeps connected as L is connected and $e(N, d)$ is connected. If the resilience is protected, then $\forall v_i \in L$ can reach d, or can reach v_N (because v_N is the only node that can reach d). Since v_N is chosen arbitrarily, for every $v_i, v_j \in L (i \neq j)$, v_i should be able to reach v_j, or there must be a traversal which protects overall node visiting after arbitrary failures, which conflicts with lemma (B.1). Proof is done.