IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON ARTIFICIAL INTELLIGENCE IN CYBERSECURITY

Received September 13, 2019, accepted September 30, 2019, date of publication October 7, 2019, date of current version October 16, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2945777

A Rerouting Framework Against Routing Interruption for Secure Network Management

MENGMENG HE[1], MINGCHUAN ZHANG[1,3,4], XIN WANG[2], JUNLONG ZHU[1], RUXI PENG[4], AND QINGTAO WU[1]

[1]College of Information Engineering, Henan University of Science and Technology, Luoyang 471023, China
[2]Laboratory of Applied Brain and Cognitive Sciences, School of Business and Management, Postdoctoral Research Station, Shanghai International Studies University, Shanghai 200083, China
[3]Henan Qunzhi Information Technology Company Ltd., Luoyang 471003, China
[4]Guangzhou Xiangxue Pharmaceutical Company Ltd., Guangzhou 510663, China

Corresponding authors: Mingchuan Zhang (zhang_mch@haust.edu.cn) and Xin Wang (wangxin@shisu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant U1604155, Grant 61602155, and Grant 61871430, in part by the Scientific and Technological Innovation Team of Colleges and Universities in Henan Province under Grant 20IRTSTHN018, in part by the basic research projects in the University of Henan Province under Grant 19zx010, and in part by the Science and Technology Development Programs of Henan Province under Grant 192102210284.

ABSTRACT In this paper, we consider the problem of routing interruption, which has received significant interest in recent years. To solve this problem, rerouting is an effective way. However, how to design an effective secure and applicable rerouting algorithm remains a significant challenge. To this end, we propose a fast rerouting framework for routing interruption. Our framework consists of two parts: (1) Diagnose routing interruption. We determine the location of interruption by directly interrogating the control plane IPv6 stack. (2) Implement fast rerouting. We propose the rerouting algorithm to make the router match and reroute all prefixes affected by interruptions. This paper updates the path only by adding a forwarding rule, thus the interruption recovery time is reduced. We conduct a comprehensive evaluation for our rerouting framework. Experimental results show that our method is better than FCP, OSPF and DLF in speed and accuracy.

INDEX TERMS Internet reliability, rerouting, routing path, tag.

I. INTRODUCTION

As the Internet serves more and more applications [1]–[5], various cybersecurity problems are constantly exposed, which lead to a series of network failure [6], [7]. The most common problem in network failure is the routing interruption problem [8]. The interruption may lead to a large amount of finance or prestige loss. Reference [9] shows that the average loss of downtime caused by interruption is 8,000 dollars per minute. Whereas, this loss reaches to 100,000 dollars or more for e-business or search engine sites. The routing interruptions are due to many reasons [10]–[13], such as interface failures [10], software defects [11], and attacks [12], [13]. Reference [14] shows that even in a well-maintained network, the interruptions cannot be completely avoided, so it is more urgent to recover interrupted services in a short period of time. The fundamental problem of interruption recovery is how to replace a damaged path with a new path. One of the main methods used by the current network is rerouting. Rerouting techniques [15]–[26] are proposed to solve the interruption problem by enabling routing protection.

IP routing protocols such as OSPF have slow routing convergence processes against failures because they are global and reactive. In this context, in order to make the routing unaffected by the interruptions, a fast rerouting method is proposed. The fast rerouting method does not have to wait for the convergence of the routing protocol, but can quickly switch traffic to the backup next hop or backup path. Some previous rerouting works that do not require tags and choose backup next hop by taking information from traditional IP packet forwarding [17], [18]. Although the methods cited above do not require tags, a high fault resilience cannot be guaranteed. To improve the fault resilience, we use tags in rerouting algorithms since the tag-based approach provides better protection performance in rerouting techniques [19].

The associate editor coordinating the review of this manuscript and approving it for publication was Nan Cheng.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/

In [20], [21], these methods ensure that the routing of the k-color tree cannot be affected by $(k-1)$ link failures. However, complete protection is only provided when the network topology is k-connected. The Failure-Carrying Packets (FCP) [22] reduces the overhead of each packet by using tags to indicate the set of faults in the packet. The tag is dedicated to directional links and therefore FCP only makes sense between adjacent nodes. However, k-connected or adjacent nodes may be interrupted when multiple links fail simultaneously. For this reason, the design of the secure rerouting methods is necessary. To the best of our knowledge, the effective rerouting method, which employs tags and is independent of network topology, has barely been investigated.

To fill this gap, we propose a fast rerouting framework to locate the router interruption and rerouting quickly. Because we only need to encode the first few positions of the path, we do not make requirements on the topology. We implement rerouting by updating the pre-computed routing path, and our method is no limit to the nodes and links, so the applicability of our method is better in the same network environment. We extend a few disjoint failover paths to reduce the number of paths affected by interruption and packet loss rate for improving the protection performance of the rerouting.

In our proposed framework, a diagnosing routing interruption method is presented in the first part. We use an active probing technology that extracts IPv6 fragment identifiers from routers efficiently, and then the router interruption is inferred by the fragment identifier continuity. In this part, the adaptive active probing technology is used to identify the router interruption, and the IPv6 alias resolution technology is used to reduce the router interfaces. The second part implements fast rerouting of packets affected by the interruption. We pre-calculate the routing path of the packet and accelerate data plane updates by a two-stage forwarding table. At the same time, we propose a rerouting algorithm that is used to make the router match and reroute all prefixes affected by interruptions.

Our contributions are listed as follows:

- A fast rerouting framework for network interruption is proposed. It includes a diagnosing routing interruption method and a fast rerouting method.
- A method for enabling an existing router to quickly recover a connection in the event of an interruption. The router accelerates data panel updates by a two-stage forwarding table that containing traversal path information and forwarding rules.
- An algorithm for fast rerouting is proposed. This algorithm gives the router the flexibility to match and reroute all prefixes affected by the interruption.

This paper is organized as follows. Section II provides related work for the study. The rerouting framework is provided in detail in Section III. Section IV introduces the rerouting algorithm. Section V presents the experimental result. Finally, our conclusion is presented in Section VI.

II. RELATED WORK

A meaningful previous work is to check the general network resilience, especially the Internet interruption problem. Trinocular [23] and ThunderPing [24] send data plane probes continuously from measurement nodes to the edge of the network to detect accessibility problems, while [25] puts the BitTorrent node into crowdsourced measurement. These methods demonstrate that accessibility is related to natural disasters and political events. However, these methods require a lot of detection traffic and can not identify specific routers as the root cause of failures.

We can recover the routing by fast fault detection [26] in a short time, so they are called routing protection or fast rerouting (FRR). Some FRR methods are based on MPLS [27] or pure IP [28]. In [29], the IP Fast Rerouting (IPFRR) method is designed. IPFRR has two principles of precomputed detours and local rerouting. Local rerouting means that only notify routers that are directly adjacent to the fault, thus alleviating the time-consuming problem of the global flooding step of the fault information on the Interior Gateway Protocols (IGP) recovery process. In addition, the IPFRR mechanism is proactive, because detours are calculated before any failure occurs, the router can immediately switch to the alternate path in the event of a failure. Nelakuditi *et al.* [18], [30] use the packet's ingress interface to select the appropriate next hop to bypass the failures. These studies focus on single-node and single-link failures. At the same time, these methods can protect the routing from any single link failure only when the network topology satisfies certain conditions.

Equal cost multipathing (ECMP) [31] is a method associated with fast rerouting. It computes multiple paths with the same cost for each source/target pair and handles failures on the path by sending packets on the alternate path. But the same cost path does not always exist between any two nodes [28]. In [17], a scheme for finding loop-free alternate paths is proposed. The routing from source S to destination D is considered. If the neighbor node X of S satisfies $d(X, D) < d(X, S) + d(S, D)$, X can be used as an alternate path, but the node may not be able to find such a neighbor. So these methods cannot ensure a high fault resilience.

Some rerouting methods have limited scope of application. In [22], a new routing model is proposed. The goal of this model is to completely eliminate the convergence process rather than reducing the convergence time. For this reason, [22] proposes a FCP technology that allows packets to autonomously discover working paths without the latest complete state of the router. The FCP reduces the overhead of each packet by using tags to indicate the set of faults in the packet. However, the tag is dedicated to directional links and therefore FCP only makes sense between adjacent nodes. In [21], an IP-based fast rerouting mechanism based on rooted arc-disjoint spanning tree is developed, which guarantees the recovery of $(k-1)$ link failure from a K-edge connected network. This method provides excellent scalability because it can construct disjoint spanning trees in the sub-quadratic

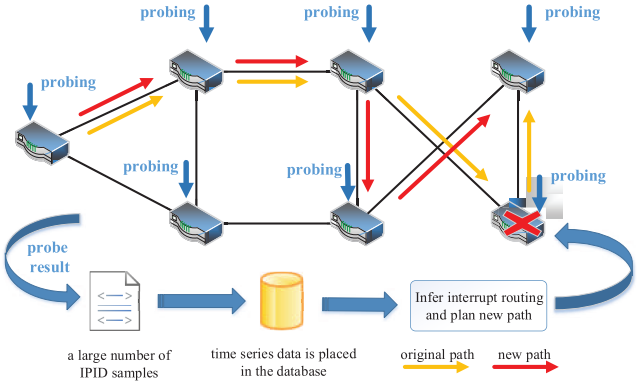


FIGURE 1. Rerouting framework workflow.

time of the size of the network. Reference [32] proposes a method for separating multi-path routing and quick recovery in an IP network. This method guarantees recovery from any two link failures. Reference [32] develops three routing methods that use trees to limit routing table entries to up to four for per node, so the packet overhead is very small. The path from source to destination in the tree is a link that does not intersect. Packet routing is based on the input interface and destination address of the received packet. Due to its computational complexity, it is not practical to employ a link-independent tree that recovers from a large number of failures. Compared with these methods, our method has better performance in time and packet loss rate.

In order to determine whether the router is interrupted, we use a method of directly interrogating the control plane IPv6 stack on the router [33], which is more accurate than the prior technology. We achieve faster failover and reduce packet loss by pre-computing and spreading a few disjoint failover paths. At the same time, the extended path reduces the number of paths affected by the interruption, so the protection performance of rerouting method is improved. By inferring the interruption from a single point, the connection is restored in a matter of seconds, thus our method is more flexible in terms of inference interruptions and input sources.

III. REROUTING FRAMEWORK

We propose a rerouting framework that consists of two parts: i) Diagnosing routing interruption. ii) Fast rerouting. As shown in Figure 1. First, we use prober to probe the router for generating a large amount of sample data, then Algorithm 1 is used to infer the interrupted router by analyzing these samples (part A). Next, we execute the rerouting algorithm (Algorithm 2) to plan a new path when the interrupt is detected (part B). We embed the packet tag that contains pre-calculate the path of the prefix and the alternate next hop in the incoming traffic to accelerate data plane updates.

A. INFERENCE ROUTER OUTAGE

We cite the method of diagnosing routing interruption in [31]. There are two main steps: probing and reboot inference. In this method, we continuously probe each router to construct a time series of IP identifier (IPID) values from segmented IPv6 response packets. Finally, the discontinuities in this series are used to infer the router restarted some time during the polling interval. We define these restarted routes as interruptions.

1) PROBING

We utilize the active probing technology [34] to detect router interface state. We use an optimized prober. Firstly, the prober periodically samples the interface and requests a single fragment response on the interface of each monotonic sequence response. Then we record the value of the IP identifier (IPID) [35] field of prober received on the same interface. If the value of this time is smaller than the previous one, the prober determines router whether still assigning the IPID value from the counter by performing six further probes. If the interface that sends the monotonic sequence remains silent, our prober will continue to request a single segmented response that up to two hours. If the interface is still silent, the prober schedules the interface for less frequent probes. The prober is cycled that starts a new round of detection immediately after the end of the previous round.

2) REBOOT INFERENCE

Our prober generates a large of IPID sample every day. In order to perform range-based queries more efficiently, we put all time series data (including IPID and router updates) in the Apache Cassandra NoSQL database [36], and the target and the timestamp are primary keys.

Algorithm 1 describes a method for inferring a router restart based on IPID time series. To avoid false positives because of the interface having a large natural IPID speed, Algorithm 1 calculates an expected value E[spin] of the IPID changes for a weighted moving average of given historical IPID speed. If the sequence is not random and the discontinuity is greater than an order of magnitude of the expected change, we infer that it is a cyclic restart. The active probing method can infer the interruption window - the period of time when the router is restarted.

B. FAST REROUTE

After we execute the diagnosing routing interruption method, we get that router 5 (Abbreviation R5, the same below) is interrupted. Then the router 1 will perform the rerouting method to infer the affected prefix and update the alternate next hop to the primary next hop of the corresponding router. Finally, we reroute the affected prefix. Because we embed the path, forwarding rule and other information into each packet by using the tag, we do not need to conduct path lookup and other operations when the interruption occurs, and can directly add a forwarding rule in the tag to quickly realize rerouting. And we can achieve faster failover and reduce packet loss by pre-computes and extending several disjoint failover paths.

Algorithm 1 Findoutage(id[ ], tx[ ])

```
v ← 0
min_id ← 0
reboots ← { }
for i ∈ |id| do
    v' = (id[i] − id[i − 1])/(tx[i] − tx[i − 1])
    v = 0.8v + 0.2v'
    E[spin] = (tx[i] − tx[i − 1]) ∗ v
    if id[i] < id[i − 1] then
        if (|id[i] − min_id| < 2^16) ∧ (id[i] > 2^16) then
            Cyclic
        else if rand_seq(id[i, i + 10]) then
            Random
        else
            reboots ← reboots ∪ tx
        end if
    else if id[i] > id[i − 1] + E[spin] ∗ 10 then
        if rand_seq(id[i, i + 10]) then
            Random
        else
            min_id ← id[i]
            reboots ← reboots ∪ tx
        end if
    end if
    if id[i] < min_id then
        min_id ← id[i]
    end if
end for
return reboots
```

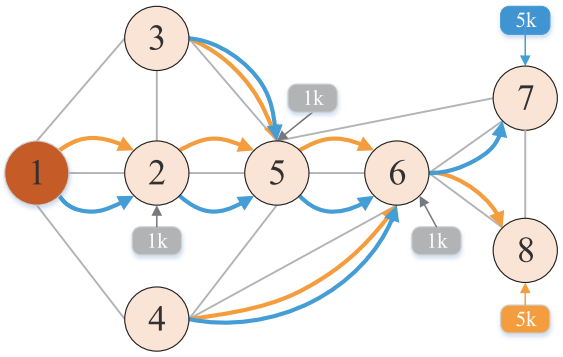


FIGURE 2. Router forwarding paths.

1) OVERVIEW

Firstly, we need to constantly pre-compute the backup next hop of the router for using in the event of an interruption. We perform this calculation for each prefix and take into account any links on the corresponding router. We consider the operator's policies (for example, cost models and peer types) and performance criteria (for example, we avoid rerouting large amounts of traffic to low-bandwidth paths.) to select alternate paths. And we extend disjoint failover paths to achieve faster failover. For example, in Figure 2, the R1 selects R3 or R4 as the alternate next hop for rerouting the 10k prefix advertised by R7 and R8 when the R5 fails. However, R4 can bypass R5 directly to the rear router, so we choose R4 as a backup for R5.

When the router performs a rerouting operation, it updates data plane rules based on each prefix. The router may need to update forwarding rules with thousands of prefixes, so rerouting operations are slow. And the forwarding rule is implemented in the data plane. So we accelerate data panel updates are helpful for accelerating rerouting. Our method accelerates data plane updates by packet tags. We embed the tag of the data plane into each incoming packet. The router accelerates data plane updates by a two-phase forwarding table. For each path depth, we have a primary next hop and an alternate next hop. They are embedded in the incoming packet by tag in the first phase of the two-stage forwarding table. Namely, each tag contains a list of router paths that are traversed, and the backup next hop that using if any of the router failed.

After we know that R5 is interrupted by Algorithm 1, the R1 running rerouting algorithm that can quickly identify a set of possible affected prefixes. Then, all affected traffic pretreated redirect to the pre-calculated backup next hop. We mainly use a single forwarding rule that matches the data plane tag installed on the packet. This forwarding rule is embedded in the incoming packet by tag in the second phase of the two-stage forwarding table. When rerouting is required, we only need to add a high priority forwarding rule to the forwarding table. We prove that as long as the inference is accurate enough, our method is safe. When the interruption information is extracted completely and the new routing information is installed in the routing forwarding table, the router forwarding rules that we installed will be deleted, meanwhile, the rule will back to the original router forwarding rule.

2) FORWARDING RULES

Our rerouting method using packet tags. The router accelerates data plane updates by a two-phase forwarding table. The first phase contains rules that tagging packet of traversal. The tag comes with two pieces of information: (i) The routing path they are currently forwarding; (ii) The next hop (the alternate next hop to use when the router is interrupted, or the primary next hop to use when the router is running.). The second stage forms packet forwarding rules based on the first stage tags. By matching the tag, the router gets the packets that pass through the given router and reroutes them to the next hop of pre-computation.

For example, we describe the rules in the router forwarding table in R1. Figure 3 shows the tag returned by the rerouting. The rules of adding tags in the first stage of the forwarding table are consistent with the router paths used. The forwarding path of the prefix in R8 is (1, 2, 5, 6, 8). The rule is as follows,

$$match(dst\_prefix : in\ R8) \gg set(tag : 10011\ 11100).$$

The first part (red part) of the tag is used to identify the routing path. It maps a specific subset of the routing path to

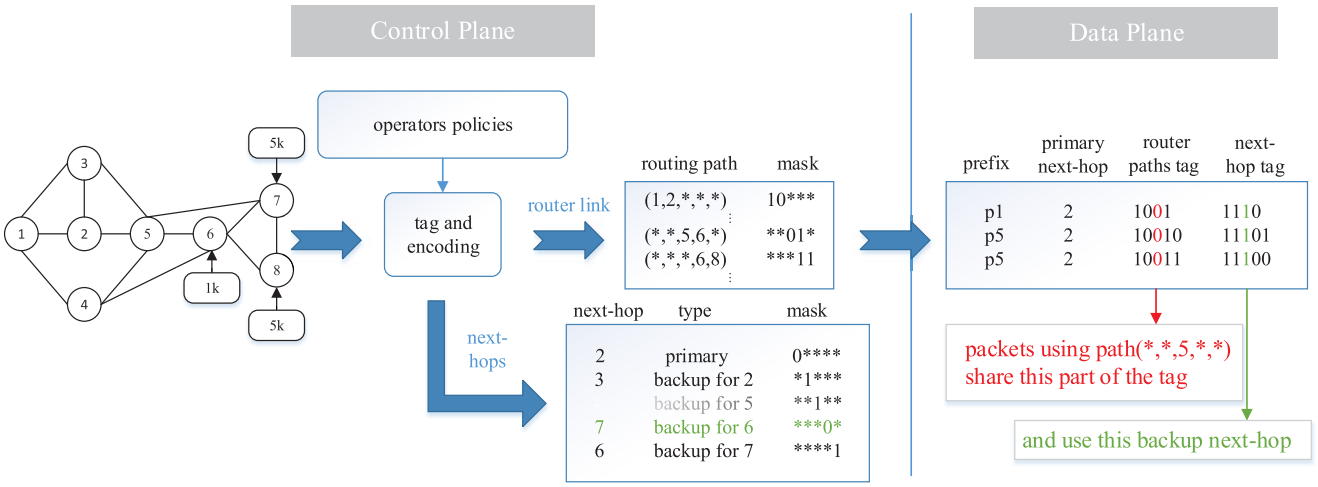


FIGURE 3. The tags structure for rerouting.

a given position. The first position indicates R1. As shown in Figure 3, this bit is set to 1. Similarly, the second and third bits indicate R2 and R5, and so on. The second part (green part) of the tag is used to encode the primary and alternate next hops. The first bit is used to identify the primary next hop, and the second and third bit are used to mark the backup next hop when R2 and R5 fail respectively. The second part of the tag is designed to match the traffic that may be redirected to a different next hop. Details are shown in Figure 3.

Before the R5 failure, the second phase only involves forwarding rules that are consistent with the routing path. Especially,

$$match(tag : 1 * * * *\ \ 1 * * * *) \gg fwd(1).$$

But when R5 fails, in the second phase, we will add a separate high priority rule instead of modifying it completely in the first phase.

$$match(tag : * * 0 * *\ \ * * 1 * *) \gg fwd(4).$$

The added rule reroutes the affected 6k prefixes traffic by using the tagged structure. The regular expression in the tag matches all the packets. For example, R5 is the third position in the routing path (i.e., the tag begins with **0**.); and its backup next hop is R4 (i.e., **1** as the end of the tag.). This includes traffic with the prefix of router 6, 7 and 8. Note that, one rule is sufficient because R5 only appears in one location before the failure in our example, so no other locations need to be considered.

3) ENCODING SCHEME

In this paper, the tag in the first stage of the forwarding table is divided into two parts: The first part is encoding the routing path for passing the packet, and the second part is encoding the next hop for all routing paths to make the packet arrive the destination. As shown in Figure 4.

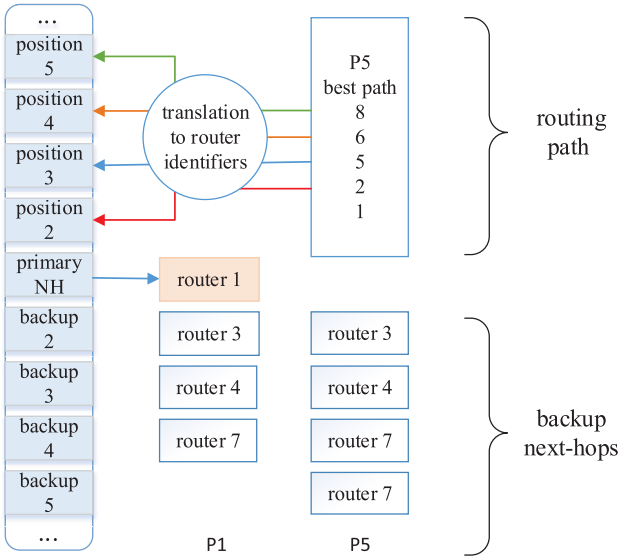


FIGURE 4. A router embeds a tag into incoming packets.

a: ENCODING ROUTING PATH

First, the tag encodes the routing path for each packet. We consider the routing path associated with the optimum routing and store the location of the router in this path for each prefix. According to the requirement of users, different routing protocol can be used to select the optimum routing. The first hop in any routing path is indicated as the primary next hop, so position 1 is not modeled. We have different routing path identifier for each router neighbor. By observing the routing path, many routing paths have very few prefixes, so we allow each prefix to be updated. For longer paths, we only need to encode the first few positions of the path, because the intermediate node may know the alternate path behind it.

For the rest of the paths, we encode the paths that traverse their maximum number of prefixes. For this purpose,

we use adaptive bits for each route location. Each location is achieved by a different set of bits, whose length is determined by the number of routers in the position. We map all routers in each location to a specific value of the corresponding bit set respectively. Therefore, the number of bits required for all values in this location is equal to the size of the set.

b: ENCODING BACKUP NEXT-HOPS

Figure 4 shows the path tag for the 5k prefix of R8 and the 1k prefix of R6. The second part of the tag is used to identify the primary next hop and the alternate next hop for which each routing path is encoded. For each prefix P, the first hop in the routing path of P is extracted as the primary next hop. For example, the primary next hop of the 5k prefix of R8 is R1, and the routing path is (1, 2, 5, 6, 8). To prevent the interruption of the R5, we can choose R4, because it does not need to go through R5 to reach R6. When we update the path due to interruption, if the same route appears in the path (i.e., P5 in Figure 4), we will directly merge the two hops. If two adjacent hops are not connected, we will add one hop.

The bits are divided in two parts of the tag. There is a basic trade-off between the amount of routing paths and the amount of alternate next hops that any router can encode. We allocate more bits to indicate the routing path, allowing the router to cover more failures. At the same time, we allocate more bits to indicate next-hop, allowing the router to reroute traffic to a larger number of backup paths.

Assuming we use the 48-bit tag of destination MAC for encoding, 30 bits are used to encode the backup next hop, because our experiments show that the 18-bit are used to encode route path is the most efficient. If our framework supports failures to depth 3, then the allocated bits for backup next hop need to be divided into 4 parts (1 primary + 3 backup next hop). Each depth is reserved for 7 bits and the other two bits can be assigned to code route path. And we converted to $2^7 = 128$ possible next hops. If you want to account for a failure of depth 4, the number of next hops is $2^6 = 64$.

C. DEPLOYED

In Section V, we show that how to use the target MAC to tag incoming packet similar to [37], [42]. The destination MAC is a valid ''tag carrier'' and provides a large number of bits (48). Just like any IP router, it can be easily removed by rewriting to the actual next hop MAC address in the second phase of the forwarding table. Namely, we can delete the forwarding rules used by the previous rerouting. Our framework deployed to the router only requires a software update, because many router platforms support two-phase forwarding table. We mainly deploy our framework by inserting the SDN switch and controller between the router and its peers. This deployment is similar to SDX flat [37], [42]. See part G of Section V in detail.

We achieve a conservative approach to infer the router interruption and choose the backup path. Nonetheless, we cannot ensure complete accuracy of this method because the inference is based on the IPID portion of the packet, and network conditions (such as network congestion, etc.) can cause inference errors. In these cases, the router will be dropped when the router may reroute traffic to the interrupted router, multiple routers may create interdomain loop or the destination router is interrupted. However, our results show that the router that is interrupted is rarely selected as the next hop in the routing path.

Algorithm 2 Rerouting Algorithm

```
input failRNum
router←[ ]
n←0
v←0
m←1
```

$$Allpath = \begin{Bmatrix} path1 & path2 & \cdots \\ backuppath1 & backuppath2 & \cdots \end{Bmatrix}$$

```
path ←[ ]
backuppath ←[ ]
[h,l]=size(Allpath);
if failRNum ⊆ router then
  for Allpath_{1m}; m < l; m++ do
    path = Allpath_{1m};
    backuppath = Allpath_{2m};
    for n; n < path.length(); n++ do
      if failRNum = path[n] then
        path[n] = backuppath[n];
        v = v + 1;
      end if
    end for
  end for
  if v = 0 then
    Mark failRNum is an infrequent probe
  end if
else
  Delete router failRNum
end if
return Allpath
```

IV. REROUTING ALGORITHM

This section introduces the theme of the rerouting. As shown in Algorithm 2. After we detect the interrupted router, we start running the rerouting algorithm. The rerouting algorithm is used to identify the path affected by the interruption and update it for rerouting.

If the interrupted router is not in the router set of all routing paths, we will remove this router from the interrupt set temporarily. We need to traverse all routing paths to determine whether the path contains the interrupted router. We replace the primary next hop at this location with the alternate next hop if the path contains interrupt router, otherwise reduce the detection frequency of this router in the detection phase, even remove this router from the interrupt list.

Our rerouting algorithm quickly identifies all affected prefixes and updates their paths. The path traversed and the alternate next hop in the algorithm corresponds to the information stored in the tag. For each prefix, we traverse its path to determine if it has been interrupted. We add a forwarding rule with higher priority in the second stage of the forwarding table if the path needs updating. The newly added rule makes the alternate next hop of the interrupt routing the primary hop. Such as 11-16 lines of algorithm 2. After the packet forwarding rules are changed, the router forwards directly without waiting for other prefixes. When the added forwarding rule is executed, we will delete this rule and switch back to the original forwarding rule. If the interrupted router does not affect the forwarding of all current prefixes, we want to reduce its detection frequency to save network traffic consumption.

V. EVALUATION

We evaluate the rerouting framework implemented in Python that combines active probing technique with rerouting method. First, we describe the data set used. Then, we evaluate the active detection technology and the efficiency of rerouting framework data plane coding. Next, we compare the time, packet loss rate with FCP [22], Distributed Least Flow (DLF) [38] and OSPF [39]. Finally, we compare the path stretch performance with FCP, IT [32] and ADST [21].

A. DATASETS

In this work, we use a macro Caida IPv6 traceroute topology data [40] to make a set of larger IPv6 router interface as probe target, which requires a more adaptive detection method. The IP topologies of IPv6 Internet are included in the IPv6 topological data set.

B. ACTIVE PROBING

The active probing method can infer the interruption window - the period of time when the router is restarted. The results show that 749,451 restarts were inferred from January 18, 2015 to May 30, 2017, involving 59,175 (40%) of the 149,560 response routers. Most detect router breaks are less than 2 hours (Figure 5) and restart less than twice (Figure 6). Figure 5 shows the maximum and minimum interrupt window lengths for each address. At least half of the maximum interrupt windows are less than 31 minutes, while another 22% have at least two hours and only 4% more than 24 hours. Figure 6 shows that the number of router interruption inferred each router. We conclude that 71% have less than ten interruptions, 53% have four interruptions, and 24% have one interruption in our probing.

Then in these single-point failure routers, we can locate the ::1 addresses of the prefixes of these routers to 90 different countries via Maxmind [20]. The results show that interruptions in any given country are no more common than in another country.

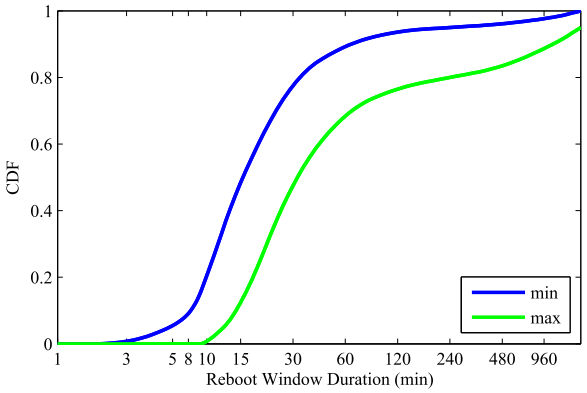


FIGURE 5. CDF of minimum and maximum outage window lengths measured.

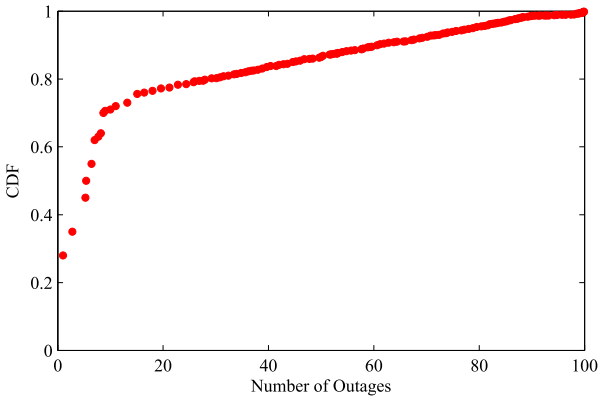


FIGURE 6. CDF outages per router for the routers that experienced an outage.

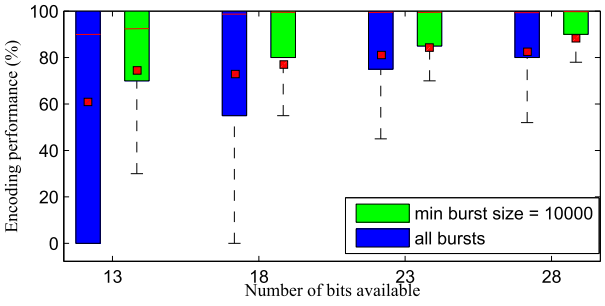


FIGURE 7. Only 18 bits are available for routing path encoding.

C. ENCODING EFFECTIVENESS

We evaluate the encoding scheme by performing a match on pre-configured tags to count the number of prefixes that can be effectively rerouted in the data plane. For each interruption burst, the coding performance is reflected by the score of the predicted prefixes, and the prefixes can be re-routed by the encoding scheme. The number of bits assigned by the partial route path of the tag determines performance.

The 18 bits of the tag is assigned to the route path portion to reroute the predicted prefix (about 98.7%). Figure 7 shows the

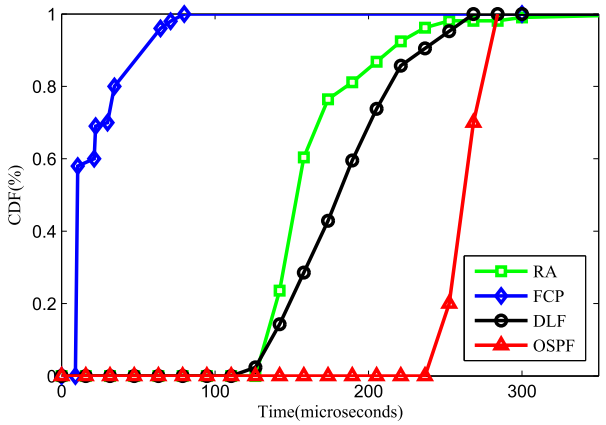


FIGURE 8. Our algorithm completes the update within 300 milliseconds.

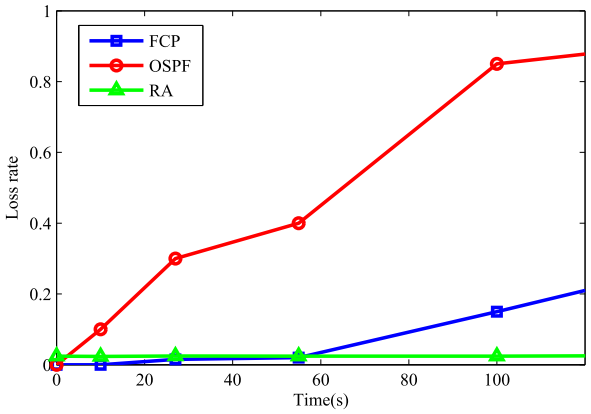


FIGURE 9. The relationship between time and loss rate.

encoding performance (on all bursts), which is represented by a function of the number of bits of the routing path reserved by the tag. In the figure, each box shows the quartile range of encoding performance: the whisker represents the 5th and 95th percentiles, the red line in the box indicates the median value and the point represents the average value. The experimental results show that the encoding performance increases with the increase of bits allocated to routing path coding. These results indicate that the compression of the encoding scheme is valid, and we encode most of the relevant routing links successfully. Figure 7 shows that large bursts with at least 10k withdrawals, the encoding performance is better. We encode the router links with the biggest prefixes with the highest priority and traverse them (And in the event of a failure, it can lead to a big burst).

In addition, if routing devices are connected to a large external We can increase the number of alternate next hops by decreasing the number of router hops encoded neighbor (such as IXP [41]), our method still works. (for example, up to depth 4 instead of 5).

D. REROUTING SPEED

In this section, we evaluate the runtime of the rerouting algorithm. In this test, we use 200 paths to test the time of our rerouting algorithm. Meanwhile, we make a comparison with FCP, OSPF and DLF. Figure 8 shows the CDF function of the time for these methods.

It is observed from the figure that the FCP time is slightly smaller than RA. The FCP is measured on a 2GB RAM with 3GHz Intel Pentium processor, and its recalculation time for all topologies is less than 100 millisecond. But the rerouting time of FCP is the time to update one link, the rerouting time of RA is the time to update all prefix paths, so for the whole, our algorithm takes less time.

We mainly record the time of recalculating path after a link failure. In this paper, we perform a rerouting algorithm (abbreviated as RA) to update the path when we detect the interrupt. According to the test, our algorithm completes the update within 300 milliseconds. Figure 8 shows that the RA method also takes less time than DLF and OSPF.

Because our method only needs to add a high-priority rule that can directly modify the routing path after an interruption, and we can directly forward affected packets after adding rules, thus our method reducing the time of rerouting after interruption occurs. We reroute all predicted prefixes with little data plane update, and the number of data plane updates depends on the number of failed routing links reported by the interrupt detection method. Each backup hop requires a data plane update for each reported link.

E. PACKET LOSS

In this paper, the packet will be dropped when the router reroutes to the interrupted router or multiple routers form an inter-domain loop. We test the relationship between rerouting time and loss rate and we compare it with FCP and OSPF. Figure 9 shows the comparison of the packet loss rate for FCP, OSPF, and RA.

By changing the rate of OSPF sends packets to neighbors, and measuring the impact on the score of the delivered packets. The figure shows that the OSPF packet loss rate is greater than the FCP and RA. For FCP, with the detection rate increases, the routing time decreases and the number of lost packets decreases because the FCP reacts more quickly to failures. In our method, when the destination route is interrupted, the packet will be dropped because it cannot be delivered. The time of our method has little effect on the packet loss rate, because our packet loss is mainly caused by the interruption of the destination node, and our method responds faster to the interruption.

F. PATH STRETCH

After a failure, the path that we rerouted is not necessarily the shortest path, which will result in a tensile loss. And stretch is defined as the value of the traversal hop count divided by the shortest working path hop count. Figure 10 shows the CDF of path stretch under several different rerouting schemes. We compare the path stretch of FCP, ADST, IT and RA.

For the ADST and IT methods, we choose a better performance for them: in the absence of a failure, the default route

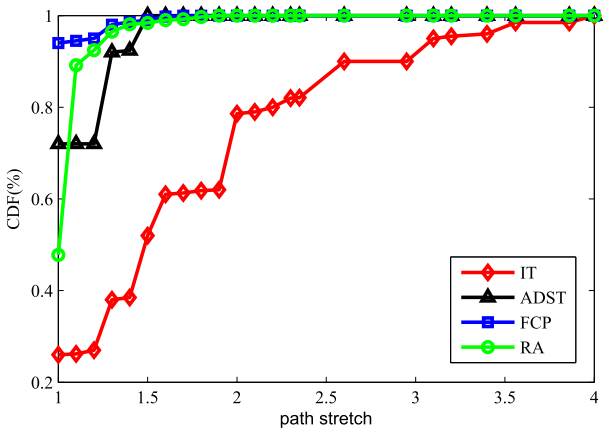


FIGURE 10. CDF for path stretch of different reroute schemes.

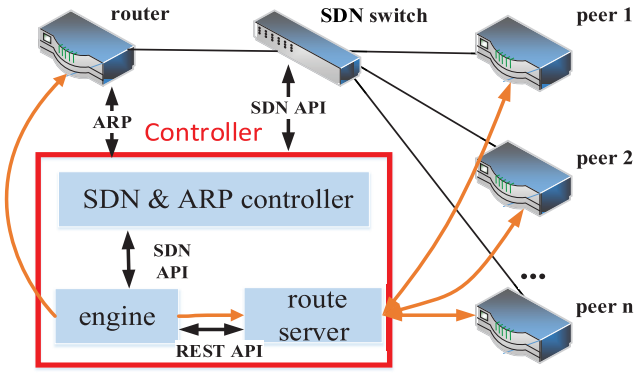


FIGURE 11. Alternative deployment scenario.

is located on the shortest path first tree (SPF) rooted in the target. The results show that the path stretch performance of our method is not lower than other methods.

G. CASE ANALYSIS

Because our method only has one hardware requirement, the two-stage forwarding table, it can be easily deployed on the current router by a simple software update. In this section, we show the benefits of deploying the framework of this paper. In our alternative deployment scenario, we insert an SDN switch and a controller (red box) between the router and its peers at the control plane and data plane levels. As shown in Figure 11. The setup is similar to the SDX platform [37], [42], which allows the deployment of this framework on any router that supports ARP and BGP, i.e., almost any router.

After receiving the rule updated from the router peer, the controller allocates a 48-bit tag according to the encoding method. The controller writes router program to embed the data plane tags into the target MAC field in the incoming packet header, using the same technique (OSPF and ARP) as in SDX [42]. It routes traffic according to the tag and rewrites the target MAC address with the actual next hop by programming the SDN switch. The two-phase forwarding table in the framework spans two devices: the router (first phase) and the SDN switch (second phase). When an outage is detected, the controller runs the rerouting algorithm and provides the data plane rule that has been updated to the SDN switch for rerouting traffic.

Although the active detection technique used in this paper applies only to IPv6 control plane router, the advantages of IPv4/IPv6 infrastructure sharing, so the restart events we detected are often manifested as IPv4 interrupts and exits [43]. This paper periodically probes the router and adds the forwarding rules in each input packet, so we can quickly change the routing path once a route interruption is detected. At the same time, because we only need to add one forwarding rule when rerouting, so the speed is improved. However, since our work requires planning the path for the prefix in advance, the cost is increased. In this paper, we use the active detection method to locate the interrupt, but our diagnosing routing interruption method can be replaced according to user requirements. For our rerouting method, we only need to detect the interrupt result.

VI. CONCLUSION

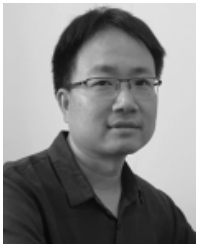
This paper has designed a fast rerouting framework. Firstly, we used the macro traceroute data to identify the IPv6 router interface, and used a new adaptive active probing technology to identify the router interruption. Then rerouting is implemented by adding a high-priority forwarding rule to the two-stage forwarding table. We have conducted a comprehensive evaluation using full-featured implementation and real router data. Our results have proved that our method is efficient in practice: coding efficiency and good packet loss rate and short running time.

REFERENCES

[1] P. Kantharaju S. Ontañón, and C. W. Geib, "Extracting CCGs for plan recognition in RTS games," in *Proc. 2nd Workshop Knowl. Extraction Games Co-Located 33rd AAAI Conf. Artif. Intell.*, Jan. 2019, pp. 9–16.
[2] K. Peng, R. Lin, B. Huang, H. Zou, and F. Yang, "Link importance evaluation of data center network based on maximum flow," *J. Internet Technol.*, vol. 18, no. 1, pp. 23–31, Jan. 2017.
[3] N. Cheng, F. Lyu, J. Chen, W. Xu, H. Zhou, S. Zhang, and X. Shen, "Big data driven vehicular networks," *IEEE Netw.*, vol. 32, no. 6, pp. 160–167, Nov./Dec. 2018.
[4] M. Zhang, M. Yang, Q. Wu, R. Zheng, and J. Zhu, "Smart perception and autonomic optimization: A novel bio-inspired hybrid routing protocol for MANETs," *Future Gener. Comput. Syst.*, vol. 81, pp. 505–513, Apr. 2018.
[5] F. Song, Y.-Z. Zhou, Y. Wang, T.-M. Zhao, I. You, and H. Zhang, "Smart collaborative distribution for privacy enhancement in moving target defense," *Inf. Sci.*, vol. 479, pp. 593–606, Apr. 2019.
[6] J. Kleinberg, "Detecting a network failure," in *Proc. 41st Annu. Symp. Found. Comput. Sci.*, Nov. 2000, pp. 231–239.
[7] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: Measurement, analysis, and implications," in *Proc. ACM SIGCOMM Conf. Appl., Technol., Architectures, Protocols Comput. Commun.*, Aug. 2011, pp. 350–361.
[8] A. Bletsas, H. Shin, and M. Z. Win, "Outage optimality of opportunistic amplify-and-forward relaying," *IEEE Commun. Lett.*, vol. 11, no. 3, pp. 261–263, Mar. 2007.
[9] Ponemon Institute. *Cost of Data Center Outages*. (2016). [Online]. Available: http://datacenterfrontier.com/white-paper/cost-data-center-outages/
[10] E. Katz-Bassett, C. Scott, D. R. Choffnes, Í. Cunha, V. Valancius, N. Feamster, H. V. Madhyastha, T. Anderson, and A. Krishnamurthy, "LIFEGUARD: Practical repair of persistent route failures," in *Proc. ACM SIGCOMM Conf. Appl., Technol., Architectures, Protocols Comput. Commun.*, Aug. 2012, pp. 395–406.

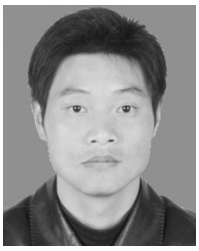
[11] A. Feldmann, O. Maennel, Z. M. Mao, A. Berger, and B. Maggs, "Locating Internet routing instabilities," in *Proc. ACM SIGCOMM Conf. Appl., Technol., Architectures, Protocols Comput. Commun.*, Sep. 2004, pp. 205–218.
[12] Y. Zhang, Z. M. Mao, and M. Zhang, "Effective diagnosis of routing disruptions from end systems," in *Proc. 5th USENIX Symp. Netw. Syst. Design Implement.*, Apr. 2008, pp. 219–232.
[13] J. N. Goel and B. M. Mehtre, "Dynamic IPv6 activation based defense for IPv6 router advertisement flooding (DoS) attack," in *Proc. IEEE Int. Conf. Comput. Intell. Comput. Res.*, Dec. 2015, pp. 1–5.
[14] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, Y. Ganjali, and C. Diot, "Characterization of failures in an operational IP backbone network," *IEEE/ACM Trans. Netw.*, vol. 16, no. 4, pp. 762–2749, Aug. 2008.
[15] J. Wang and S. Nelakuditi, "IP fast reroute with failure inferencing," in *Proc. SIGCOMM Workshop Internet Netw. Manage.*, 2007, pp. 268–273.
[16] A. Gopalan and S. Ramasubramanian, "IP fast rerouting and disjoint multipath routing with three edge-independent spanning trees," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1336–1349, Jul. 2015.
[17] A. Atlas and A. Zinin, *Basic Specification for IP Fast Reroute: Loop-Free Alternates*, document RFC 5286, 2008.
[18] S. Nelakuditi, S. Lee, Y. Yu, and Z.-L. Zhang, "Failure insensitive routing for ensuring service availability," in *Proc. 11th Int. Conf. Qual. Service*, Jun. 2003, pp. 287–304.
[19] Y. Yang, M. Xu, and Q. Li, "Fast rerouting against multi-link failures without topology constraint," *IEEE/ACM Trans. Netw.*, vol. 26, no. 1, pp. 384–397, Feb. 2018.
[20] T. Elhourani, A. Gopalan, and S. Ramasubramanian, "IP fast rerouting for multi-link failures," in *Proc. IEEE Conf. Comput. Commun.*, Apr./May 2014, pp. 2148–2156.
[21] T. Elhourani, A. Gopalan, and S. Ramasubramanian, "IP fast rerouting for multi-link failures," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 3014–3025, Oct. 2016.
[22] K. Lakshminarayanan, M. Caesar, M. Rangan, T. Anderson, S. Shenker, and I. Stoica, "Achieving convergence-free routing using failure-carrying packets," in *Proc. ACM SIGCOMM Conf. Appl., Technol., Architectures, Protocols Comput. Commun.*, Aug. 2007, pp. 241–252.
[23] L. Quan, J. Heidemann, and Y. Pradkin, "Trinocular: Understanding Internet reliability through adaptive probing," in *Proc. ACM SIGCOMM Conf.*, Aug. 2013, pp. 255–266.
[24] A. Schulman and N. Spring, "Pingin' in the rain," in *Proc. 11th ACM SIGCOMM Conf. Internet Meas. Conf.*, Nov. 2011, pp. 19–28.
[25] D. R. Choffnes, F. E. Bustamante, and Z. Ge, "Crowdsourcing service-level network event monitoring," in *Proc. ACM SIGCOMM Conf. Appl., Technol., Architectures, Protocols Comput. Commun.*, Aug. 2010, pp. 387–398.
[26] D. Katz and D. Ward, *Bidirectional Forwarding Detection (BFD) for IPv4 and IPv6 (Single Hop)*, document RFC 5881, 2010.
[27] P. Pan, G. Swallow, and A. Atlas, *Fast Reroute Extensions to RSVPTE for LSP Tunnels*, document RFC 4090, 2005.
[28] M. Shand and S. Bryant, *IP Fast Reroute Framework*, document RFC 5714, 2010.
[29] G. Rétvári, J. Tapolcai, G. Enyedi, and A. Császár, "IP fast ReRoute: Loop free alternates revisited," in *Proc. 30th IEEE Int. Conf. Comput. Commun.*, Apr. 2011, pp. 2948–2956.
[30] S. Nelakuditi, S. Lee, Y. Yu, Z. L. Zhang, and C. N. Chuah, "Fast local rerouting for handling transient link failures," *IEEE/ACM Trans. Netw.*, vol. 15, no. 2, pp. 359–372, Apr. 2007.
[31] A. Iselt, A. Kirstadter, A. Pardigon, and T. Schwabe, "Resilient routing using MPLS and ECMP," in *Proc. Workshop High Perform. Switching Routing*, Apr. 2004, pp. 345–349.
[32] A. Gopalan and S. Ramasubramanian, "Multipath routing and dual link failure recovery in IP networks using three link-independent trees," in *Proc. 5th IEEE Int. Conf. Adv. Telecommun. Syst. Netw.*, Dec. 2011, pp. 1–6.
[33] R. Beverly, M. Luckie, L. Mosley, and K. Claffy, "Measuring and characterizing IPv6 router availability," in *Proc. 16th Int. Conf. Passive Act. Netw. Meas.*, Mar. 2015, pp. 123–135.
[34] M. Luckie and R. Beverly, "The impact of router outages on the AS-level Internet," in *Proc. Conf. ACM Special Interest Group Data Commun.*, Aug. 2017, pp. 488–501.
[35] A. Bender, R. Sherwood, and N. Spring, "Fixing ally's growing pains with velocity modeling," in *Proc. 8th ACM SIGCOMM Conf. Internet Meas.*, Oct. 2008, pp. 337–342.
[36] A. Lakshman. *Cassandra*. (2008). [Online]. Available: http://cassandra.apache.org/
[37] A. Gupta, R. MacDavid, R. Birkner, M. Canini, N. Feamster, J. Rexford, and L. Vanbever, "An industrial-scale software defined Internet exchange point," in *Proc. 13th USENIX Conf. Netw. Syst. Design Implement.*, Mar. 2016, pp. 1–14.
[38] A. P. Bianzino, L. Chiaraviglio, and M. Mellia, "Distributed algorithms for green IP networks," in *Proc. IEEE INFOCOM Workshops*, Mar. 2012, pp. 121–126.
[39] *OSPF Version 2*, document RFC 2328, 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2328.txt.
[40] CAIDA. (2016). *The CAIDA UCSD IPv6 Topology Dataset*. [Online]. Available: http://www.caida.org/data/active/ipv6_allpref_topology_dataset.xml
[41] P. Mao, R. Birkner, T. Holterbach, and L. Vanbever, "Boosting the BGP convergence in SDXes with SWIFT," in *Proc. SIGCOMM Posters Demos*, Aug. 2017, pp. 1–2.
[42] A. Gupta, L. Vanbever, M. Shahbaz, S. P. Donovan, B. Schlinker, N. Feamster, J. Rexford, S. Shenker, R. J. Clark, and E. Katz-Bassett, "SDX: A software defined Internet exchange," in *Proc. ACM Conf. SIGCOMM*, Aug. 2014, pp. 551–562.
[43] A. Dhamdhere, M. Luckie, B. Huffaker, K. Claffy, A. Elmokashf, and E. Aben, "Measuring the deployment of IPv6: Topology, routing and performance," in *Proc. 12th ACM SIGCOMM Internet Meas. Conf.*, Nov. 2012, pp. 537–550.

MENGMENG HE was born in Henan, China, in February 1994. She received the bachelor's degree in software engineering from the Henan University of Science and Technology, in 2017, where she is currently pursuing the graduate degree in software engineering with the School of Information Engineering. Her research interest includes network security.



MINGCHUAN ZHANG was born in Henan, China, in May 1977. He received the Ph.D. degree in communication and information system from the Beijing University of Posts and Telecommunications, Beijing, China, in 2014. He is currently an Associate Professor with the Henan University of Science and Technology. He is also the CIO of Henan Qunzhi Information Technology Company Ltd., and Guangzhou Xiangxue Pharmaceutical Company Ltd. His research interests include bio-inspired networks, future Internet, and optimization.

XIN WANG received the Ph.D. degree in management science and engineering from the University of Posts and Telecommunications, China, in 2017. She is currently an Assistant Professor with Shanghai International Studies University. Her research interests include neuroscience, artificial intelligence, information systems, and networks.



JUNLONG ZHU received the Ph.D. degree in computer science and technology from the Beijing University of Posts and Telecommunications, Beijing, China, in 2018. In 2018, he joined the Henan University of Science and Technology, Luoyang, China, where he is currently a Lecturer with the Information Engineering College. His research interests include large-scale optimization, distributed multi-agent optimization, stochastic optimization, and their applications to machine learning, signal processing, communications, and networking.

RUXI PENG was born in Hunan, China, in July 1979. He received the master's degree in pharmaceutical engineering from Jinan University, in 2018. He currently holds the position of General Manager Assistant at the Traditional Chinese Medicine Resources Division, Guangzhou Xiangxue Pharmaceutical Company Ltd. His research interests include artificial intelligence for Chinese medicine and process optimization.

QINGTAO WU was born in Jiangsu, China, in March 1975. He received the Ph.D. degree in computer application from the East China University of Science and Technology, Shanghai, China, in 2006. He is currently a Professor with the Henan University of Science and Technology. His research interests include computer security, future Internet security, machine learning, and cloud computing.