Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 838340, 15 pages
http://dx.doi.org/10.1155/2015/838340

Research Article

Relationship-Oriented Software Defined AS-Level Fast Rerouting for Multiple Link Failures

Chunxiu Li,[1] Xin Li,[1] Ke Li,[1] Jiafu Huang,[2] Zhansheng Feng,[3] Shanzhi Chen,[1,4] Hong Zhang,[1] and Yulong Shi[1]

[1]*State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[2]*Beijing BDA Network & Information Co. Ltd., Beijing 100176, China*
[3]*Luohe Radio Administration Bureau of Henan Province, Luohe 462000, China*
[4]*State Key Lab of Wireless Mobile Communication, China Academy of Telecommunication Technology, Beijing 100083, China*

Correspondence should be addressed to Chunxiu Li; chunxiuli@bupt.edu.cn

Received 5 September 2014; Revised 4 February 2015; Accepted 15 February 2015

Academic Editor: Vincent Cocquempot

Large-scale deployments of mission-critical services have led to stringent demands on Internet routing, but frequently occurring network failures can dramatically degrade the network performance. However, Border Gateway Protocol (BGP) can not react quickly to recover from them. Although extensive research has been conducted to deal with the problem, the multiple failure scenarios have never been properly addressed due to the limit of distributed control plane. In this paper, we propose a local fast reroute approach to effectively recover from multiple link failures in one administrative domain. The principle of Software Defined Networking (SDN) is used to achieve the software defined AS-level fast rerouting. Considering AS relationships, efficient algorithms are proposed to automatically and dynamically find protection paths for multiple link failures; then OpenFlow forwarding rules are installed on routers to provide data forwarding continuity. Our approach is able to ensure applicability to ASes with flexibility and adaptability to multiple link failures, contributing toward improving the network performance. Through experimental results, we show that our proposal provides effective failure recovery and does not introduce significant control overhead to the network.

1. Introduction

With the fast development of communication infrastructure and technologies, the Internet has become the critical information infrastructure; more and more new mission-critical services are widely applied on it, such as E-bank, online games, voice-over-IP (VoIP), and video conferencing application, which have stringent demands on network reliability and availability. It is essential for the Internet to provide reliable services even in the presence of failures. Unfortunately, measurements [1, 2] indicate that link failures are fairly common events on the Internet, which can cause performance deterioration [3]. Routing protocols are expected to react fast enough to address network failures. However, current routing protocols fail to react quickly to recover from them. As the de facto interdomain routing protocol, Border Gateway Protocol (BGP) [4] experiences considerable delay in reaching convergence in the presence of network failures. During the convergence period, some destinations will not be reachable and packets through the failed link will be dropped, which may cause critical performance degradation for applications. Hence, quick recovery from link failures is the key point to guarantee network availability, which has attracted the scholars' attention for years.

Many attempts have been made in order to effectively address the interdomain routing failures [5]. One line of work is to reduce the convergence period in order to achieve a faster convergence [6–8]. Despite the fact that these solutions have been extensively studied, they are difficult to be deployed in operational networks due to their complexity or some

design flaws. Instead of shrinking the BGP convergence time to improve the network performance, another line of work aims to provide backup paths for failures [9–13]. However, these techniques mainly focus on single failure and very few address multiple failures.

As the remarkable increase of network scale makes network robustness more and more stringent, multiple failure scenarios are gaining increasing attention recently. Taking advantage of cellular graph embeddings, a novel contingent forwarding approach, called packet recycling (PR), is proposed by Lor et al. [14]. By employing extra bits in the packet header, PR tries to fast reroute the affected packets in order to eliminate packet losses under network failures. But this proposal achieves its effectiveness at the expense of the cost of extra bits and packet modifications. To achieve convergence-free routing, Lakshminarayanan et al. [15] proposed failure-carrying packets (FCP) to eliminate the convergence process completely. However, it utilizes packet labeling to handle multiple failures; considerable overhead is inevitable in the packet headers. While taking labeling free into consideration, Yang et al. [16] proposed a new routing approach named Keep Forwarding (KF). Using inport-aware forwarding, KF is designed to achieve k-link failure resilience. Using IP tunneling, Kini et al. [17] developed novel mechanisms for fast recovery from failures in IP networks. Every node in this technique requires additional three protection addresses which correspond to auxiliary graphs. Packets affected by the failed link are forwarded in a proper protection graph that does not contain the failure. Gopalan and Ramasubramanian [18] developed a routing scheme to achieve disjoint multiple routing and fast recovery in IP networks. By employing three link-independent trees rooted at every destination, the approach is able to guarantee withstanding any arbitrary two link failures. Utilizing arc-disjoint spanning trees, Elhourani et al. [19] developed an IP fast reroute mechanism to recover from multiple link failures. They demonstrated that scalable fast recovery from $(k-1)$ link failures is achievable in a k-edge-connected network. However, these approaches require either packet modifications or precomputation. In addition, they mainly focus on handling multiple link failures in the intradomain routing.

Software Defined Networking (SDN) is an emerging network architecture with separation of the control plane and data plane, which enables it to provide a good platform for innovations. As one of the most common SDN southbound interfaces, OpenFlow [20] has been applied to improve network operation and management. By employing OpenFlow technology together with traditional routing programs in the routers, Yang et al. [21] presented a framework to deal with the transient link failure. Our previous paper [22] presented software defined AS-level fast rerouting (SD-FRR), which aims to recover from interdomain link failures in one administrative domain. However, these studies only apply to single link failure.

In this paper, we propose software defined AS-level fast rerouting for multiple link failures by considering AS relationships (SDFRR-ASR) in one administrative domain. SDFRR-ASR is designed to provide fast failure recovery from various types of multiple link failures which include concurrent, continuous, centralized, and distributed link failures. Although our previous work [22] is capable of handling interdomain link failures, the main difference is that this paper mainly focuses on dealing with multiple link failures taking AS relationships into consideration. Similar to [22], the network control plane and data plane are also separated in this paper. SDFRR-ASR is a local fast reroute scheme that can guarantee fast rerouting by constructing effective policy-compliant protection paths without waiting for BGP convergence. In the solution, algorithms are proposed to find such protection paths by taking AS relationships into account, in addition to routing policies and BGP decision rules, which can improve the network performance. Upon multiple link failures, the protection paths are computed immediately after failure scenario recognition; then OpenFlow forwarding rules are installed on relevant routers to provide the packets' delivery continuity.

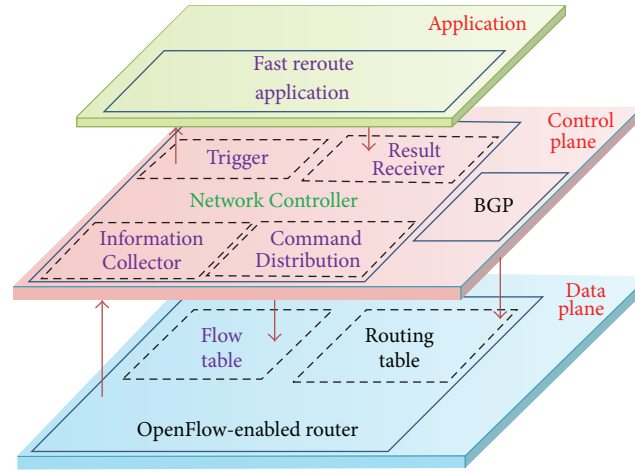


Figure 1: System architecture.

The remainder of the paper is organized as follows. Section 2 presents the system architecture of SDFRR-ASR. Section 3 describes the software defined AS-level fast rerouting application with AS relationships in detail. Section 4 exhibits the performance of SDFRR-ASR through experiments. Finally, our conclusions are presented in Section 5.

2. System Architecture

The overall structure of SDFRR-ASR is shown in Figure 1. The control plane has been separated from the data plane, which gives it the flexibility to control the network. The separation of the control plane and data plane enables the structure to have two distinguishing advantages. One is that the centralized control enables the system to control the entire network flexibly, which is suitable for handling link failures, and the other one is that the distributed control of BGP is still able to work normally. That is, the centralized control and distributed control can work normally without disturbing each other.

In the architecture, Network Controller is the core unit for the system control and management. It includes four

key components for constructing protection tunnels. Information Collector is designed to collect all the necessary network information and save it in the centralized database, which enables the Network Controller to have a global view of the whole system and the ability of centralized control. Once link failure occurs, Trigger is used to trigger the upper layer application to compute protection paths. After the application finishes its procedure, Result Receiver is responsible for receiving the results from the application. Command Distribution is responsible for sending commands to install OpenFlow forwarding rules on the underlying routers based on the received results. Besides, it is also used to repeal OpenFlow forwarding rules. Network Controller integrates the function of the above components and provides unified APIs to the upper layer applications, which enable them to implement specific applications through the APIs regardless of the details of the underlying network. It is important to note that the application developed in this paper is fast rerouting for multiple link failures in the interdomain routing with AS relationships.

In our system, OpenFlow technology is used to build the underlying protection tunnels; the BGP routing table and OpenFlow flow table are coexisting in the routers along the protection paths. In this case, the flow table has priority to forward corresponding packets according to the rule, which enables our approach to work well upon failures. As shown in Figure 1, they do not interfere with each other, which makes our scheme flexible and effective. For more details, please see our previous work [22].

3. Fast Reroute Application for Multiple Link Failures in the Interdomain Routing with AS Relationships

In this section, the fast reroute application for multiple link failures with AS relationships is explained in detail. Figure 2 shows the overall process of it. Failure scenario recognition (FSR) is responsible for accurately identifying the failure scenarios, which will be described in Section 3.1. The protection path computation (PPC) is responsible for effectively finding the optimal policy-compliant protection paths for failed links, which will be introduced in Section 3.2 in detail. The protection tunnel deactivation (PTD) is responsible for checking whether OpenFlow entries should be removed, which is introduced in Section 3.3. Note that overall the whole process, sending commands to establish or remove protection tunnels, is implemented by Command Distribution in the Network Controller, which is independent of the application.

3.1. Failure Scenario Recognition. The ultimate goal of our technique is to handle multiple link failures under different failure scenarios which include concurrent failures and continuous failures. FSR plays a fundamental role in the design of our scheme. For different failure scenarios, we need to propose different solutions to address them; thus it is crucial to identify the failure scenarios firstly and accurately.

Concurrent failures are the failures in which links fail at the same time. Upon the interdomain link failures, Network

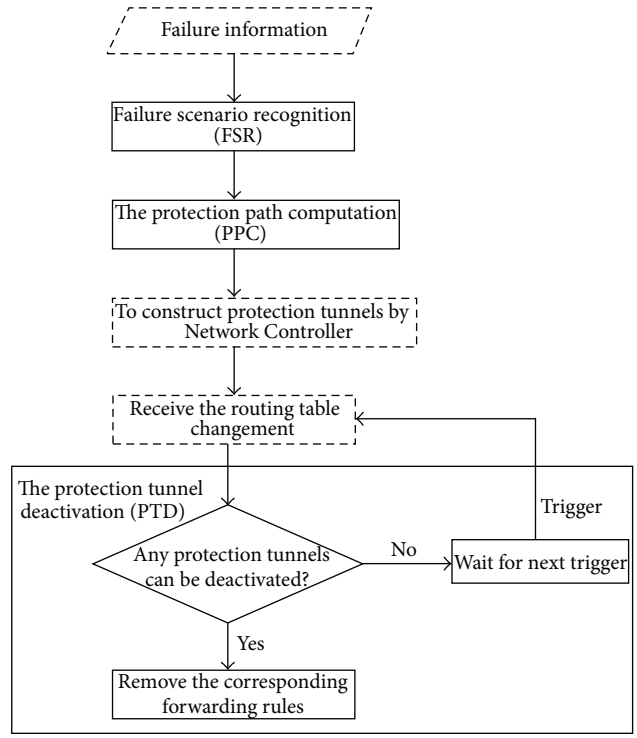


Figure 2: The overall process of the application.

Controller triggers the application to discover the protection paths in order to detour around the failed links. Unfortunately, there is an inevitable delay of the propagation time between the routers and Network Controller; thus concurrent failures may not trigger the application at the same time, which may cause the unavailability of obtained protection paths. In this case, the fast reroute application is expected to consider all the concurrent failures before computing the protection paths. We propose the approach of delaying failures handling in order to accurately handle the concurrent failures; that is, when the first failure arrives, the application should wait a configurable time T_{fsr}; then it takes account of all the concurrent failures during the interval to compute protection paths. This model can be expressed as follows:

$$L_f = \sum_{i=1}^{T} l_i, \tag{1}$$

where L_f is the set of all failed links, l_i is the set of failed links that occur in the time i, and T is the time interval T_{fsr}.

Under continuous failure scenarios, the key point is to determine whether the subsequent failures affect the existing protection tunnels. If the existing protection tunnel is influenced, it should be deactivated. After that, the application took all the failures including the failed links corresponding to the deactivated tunnels to compute the protection paths. To cope with this condition, two special types of continuous failures should be considered carefully. One is that the subsequent failure lies on the existing protection paths, and the other one is that it affects the normal use of the existing

protection path without lying on it. This model can be expressed as follows:

$$L_f = \begin{cases} l_{fl} + l_{fm}, & \text{if } l_{fl} \in T_{\text{type}}, \\ l_{fl}, & \text{otherwise}, \end{cases} \quad (2)$$

where L_f is the set of all failed links, l_{fm} is the former failed links corresponding to the affected protection tunnels, l_{fl} is the failed links this time, and T_{type} is the set of continuous failures.

3.2. The Protection Path Computation. To effectively find optimal protection paths in the face of multiple link failures, we design the process of the protection path computation (PPC), which is the key function of SDFRR-ASR. As the key factor of our approach, AS relationships are taken into account to find policy-compliant protection paths, in addition to BGP policies and decision rules, so as to guarantee the protection paths to conform to the interdomain routing. The solutions to main types of AS relationships are first discussed in Section 3.2.1, and then the detailed process of PPC is described in Section 3.2.2.

3.2.1. Discussion of the AS Relationships. AS relationships are an important aspect of Internet structure. They have great impact on the interdomain routing [23]. With the help of AS relationships, our approach is able to shorten the length of the protection tunnels based on the type of the failed link, which can significantly reduce the control overhead. Three types of AS relationships are considered in this paper: provider-to-customer relationships, customer-to-provider relationships, and peer-to-peer relationships [24, 25]. These relationships are saved in the database by administrator in practice; they can be easily got when you use them.

(1) Customer-to-Provider Relationship. In this case, the failed link is the customer-to-provider edge. Since our approach can optimize the length of protection tunnels, there is no need to detour the path of an affected prefix along the entire protection path, so the router located inside the customer AS associated with the failed link is the starting point of protection tunnel, but the endpoint of the protection path may be not the router inside the AS owning the affected prefixes. Some routers on the protection path may have routes to destinations. Under this case, two situations should be considered. One situation is that the customer AS has another provider AS except for the one associated with the failed link. The other situation is that the customer AS has no other provider AS.

(a) Customer AS Has Another Provider AS. From the customer AS's viewpoint, the providers can be used to reach any destinations. However, since our approach aims to handle multiple link failures, some affected prefixes may be unreachable from the provider AS because of the impact of other link failures.

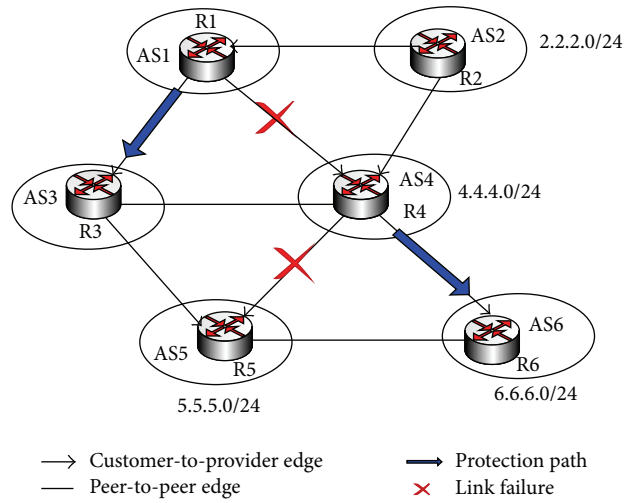


Figure 3: Provider AS has all reachable routes.

Thus, we need first to analyze whether all affected prefixes are reachable through the provider AS.

(i) All the Affected Prefixes Are Reachable from the Provider AS. If all the affected prefixes are reachable through the provider AS, then the protection tunnel will be terminated at the router located inside the customer AS that connects to the provider AS. We do not need to construct the entire protection tunnel between the router located inside the customer AS and the routers inside the ASes owning the affected prefixes. For example, AS2, AS4, AS5, and AS6 advertise prefixes 2.2.2.0/24, 4.4.4.0/24, 5.5.5.0/24, and 6.6.6.0/24 in Figure 3, respectively. Assume that links R1–R4 and R4-R5 fail, AS3 and AS4 are provider ASes of AS1, and AS5 and AS6 are provider ASes of AS4. For the failure of link R1–R4, router R1 is the starting point of the protection tunnel; since AS1 has another provider AS3, and all the prefixes affected by the failed link R1–R4 are reachable from AS3, the protection tunnel will be terminated in AS3 as shown in Figure 3. The OpenFlow forwarding rules only need to be sent at router R1. There is no need to construct the entire protection paths. In the same way, the protection tunnel built for the failed link R4-R5 is also shown in Figure 3. It should be noted that our approach can provide bidirectional protection for failed links. To briefly explain the problem, we just show the protection for one direction of the failed links in this paper.

(ii) Not All the Affected Prefixes Are Reachable from the Provider AS. If only some affected prefixes are reachable through the provider AS, then one router inside the provider AS is selected as the endpoint of the protection tunnel. The path of an affected prefix does not need to detour along the entire protection tunnel. As to the remaining affected prefixes, we construct a 2-dimensional matrix representing the connection relationship of ASes. After deleting all the connection relationships related with the failed links from the matrix, we compute the possible paths for the remaining affected prefixes; then we select the optimal endpoint routers

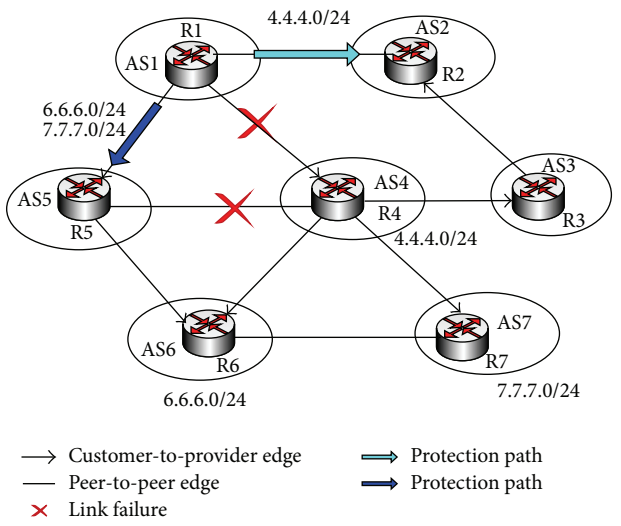


FIGURE 4: Provider AS has some reachable routes.

according to the BGP routing tables, so as to optimize the protection tunnel.

For example, consider in Figure 4 that AS4, AS6 and AS7 advertise prefixes 4.4.4.0/24, 6.6.6.0/24, and 7.7.7.0/24, respectively. AS4 and AS5 are provider ASes of AS1, AS4 and AS5 are peer-to-peer relationship, AS1 and AS2 are peer-to-peer relationship. We assume that links R1–R4 and R5-R4 fail, the affected prefixes are 4.4.4.0/24, 6.6.6.0/24 and 7.7.7.0/24 for the failed link R1–R4. Although AS5 is a provider of AS1, the best path of prefix 4.4.4.0/24 is uselessness because the failure of link R5-R4, so it only needs to build protection tunnel terminated at router R5 for prefix 6.6.6.0/24 and 7.7.7.0/24 as shown in the Figure 4. According to the approach described above, our approach builds protection tunnel R1-R2 for the prefix 4.4.4.0/24.

(b) Customer AS Has No Other Provider AS. When the customer AS has no other provider AS except the provider AS relevant to the failed link, it needs to compute the full paths between the customer AS and the ASes owning the affected prefixes and then select the earlier routers as the endpoint of the protection tunnels, so as to optimize the protection tunnels. The solution to this condition is the same as that to the remaining affected prefixes described in (ii).

(2) Provider-to-Customer Relationship or Peer-to-Peer Relationship. In these conditions, the failed link is either the provider-to-customer edge or peer-to-peer edge. The two relationships are simple and do not consider complex conditions. The solution to the two types of failed links is the same as (b) as described above.

3.2.2. The Process of the Protection Path Computation. As the key part of our approach, the protection path computation (PPC) aims to effectively find optimal protection path to detour multiple link failures. The highlight of this process is that the AS relationships are taken into account in order to meet special requirements. Besides, BGP policies and

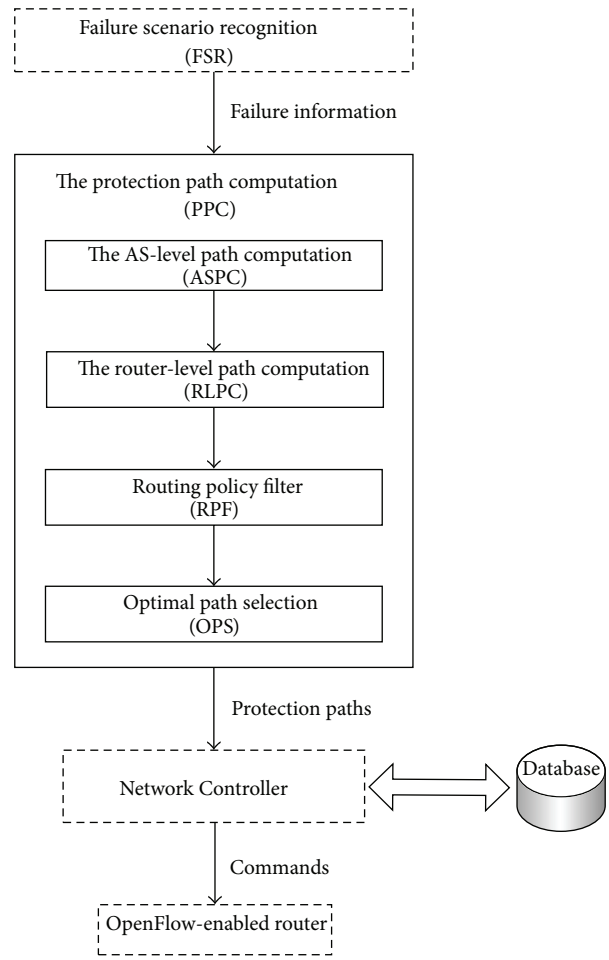


FIGURE 5: The protection path computation.

simplified decision rules are also used to help find the policy-compliant protection paths, which enable the protection paths to conform to the interdomain routing.

Figure 5 shows that there are mainly four phases in the PPC, which we refer to as "the AS-level path computation (ASPC)," "the router-level path computation (RLPC)," "routing policy filtering (RPF)," and "optimal path selection (OPS)" sequentially. Although the phases are the same as our previous work [22], some extensions have been made to meet our special requirements. Utilizing failure information from FSR, ASPC aims to find AS-level paths for all failed links. As the objective of this paper is to handle multiple interdomain link failures with the AS relationships, ASPC extends to be capable of computing AS-level protection paths by considering both AS relationships and multiple link failures, which will be described in Section 3.2.2(1) in detail. The function of RLPC is to refine the AS-level paths obtained by ASPC into the fine-grained router-level paths. In order to make the protection paths conform to the routing policies, RPF attains this objective of filtering out paths which do not conform to the routing policies. The goal of OPS is to select

optimal paths for failed links by using BGP routing tables or simplified decision rules.

(1) The AS-Level Path Computation. As our approach takes the AS relationships into account, the AS-level path computation (ASPC) is expected to accurately compute the protection paths for multiple link failures by considering them. As explained earlier, every type of the failed relationships has corresponding solutions. The ultimate goal of ASPC is to find the AS-level paths for all failed links. For each affected prefix, the obtained path is the entire AS-level path which is a sequence of ASes between the AS associated with the failed links and the AS owing the affected prefix. The basic ASPC procedure is detailed as follows.

Step 1. For the failure information received from FSR, ASPC selects the failed link to be handled at this time by considering the sequence of link failures.

Step 2. With regard to the failed link, ASPC needs to check whether any prefixes are affected by the failed link. If no prefix is affected, there is no need to compute the protection path for it; otherwise, go to Step 3.

Step 3. Considering the AS relationships, ASPC must verify the relationship of the failed link before computing the protection path for it. It checks whether the failed link is customer-to-provider edge, which is expressed as follows:

$$\text{Is\_CP} = \begin{cases} \text{true}, & \text{if (Src\_AS=Customer} \\ & \quad \&\&\text{Dest\_AS=Provider)}, \\ \text{false}, & \text{otherwise}, \end{cases} \tag{3}$$

where Src_AS is the AS which contains one endpoint of the failed link and Dest_AS is the AS which contains the other endpoint of the failed link.

If not, the protection path should be computed according to the procedure of provider-to-customer relationship or peer-to-peer relationship; then go to Step 8; otherwise, go to Step 4.

Step 4. If the failed link is customer-to-provider edge, it is expected to see whether the customer AS has any other available provider AS, which is described as follows:

$$\text{Available\_Provider} = \begin{cases} \text{true}, & \text{if } (\text{Src\_AS.provider.num} \neq 0), \\ \text{false}, & \text{otherwise}. \end{cases} \tag{4}$$

If not, it means that the customer AS has no available provider AS to use and it should consider other links; then go to Step 8; otherwise, go to Step 5.

Step 5. Utilizing the BGP routing table, the process checks whether the available provider AS has reachable routes for all the affected prefixes. If it does, the specific router in the provider AS is the endpoint of the protection path, which will be saved in the database; otherwise, go to Step 7.

Step 6. Further check whether any other failed links have not been handled. If not, the obtained protection paths will be sent to Network Controller by PPC, and the process ends; otherwise, ASPC selects the next failed link to compute the protection path; go back to Step 1.

Step 7. If there is no provider AS that has reachable routes for all the affected prefixes, ASPC continues to check whether any affected prefixes have reachable routes according to BGP routing tables. If the specific router exists, it is the endpoint of the protection path for these prefixes and will be saved in the database. As to the remaining prefixes, go to Step 8. If the specific router does not exist, go to Step 8.

Step 8. ASPC checks whether any EBGP links are available based on the AS information saved in the database. If no EBGP links are available, PPC sends the protection path saved in the database to Network Controller, and the process ends; otherwise, go to Step 9.

Step 9. Using the information of ASes saved in the database, ASPC constructs a 2-dimensional matrix $\mathbf{A}$. It represents the connection relationship of ASes, which is expressed as follows:

$$\mathbf{A} = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}. \tag{5}$$

The element $\mathbf{A}[i, j]$ of $\mathbf{A}$ represents the connection relationship between AS_i and AS_j, which can be expressed as follows:

$$\mathbf{A}[i, j] = \begin{cases} N, & \text{if } \text{AS}_i \text{ has } N \text{ connection edges with } \text{AS}_j, \\ 0, & \text{otherwise}. \end{cases} \tag{6}$$

Step 10. According to the failed links, ASPC deletes the connection relationship of ASes attached to them in the matrix.

Step 11. Based on the information of failed links and the obtained matrix, ASPC uses the depth-first algorithm to find N AS-level paths for each affected prefix. N is a configurable number that can be configured by network operators.

Step 12. The set of N paths are the obtained AS-level paths, and they will be sent to RLPC by ASPC.

The flow chart of the original ASPC is given in Figure 6.

(2) The Router-Level Path Computation. In our scheme, the protection path is constructed by distributing OpenFlow commands on specific routers, so the router-level paths are required after the AS-level paths are found by ASPC. To this end, we design the process of RLPC to refine the AS-level paths into router-level paths. As in our approach, all the necessary information is saved in the centralized database,

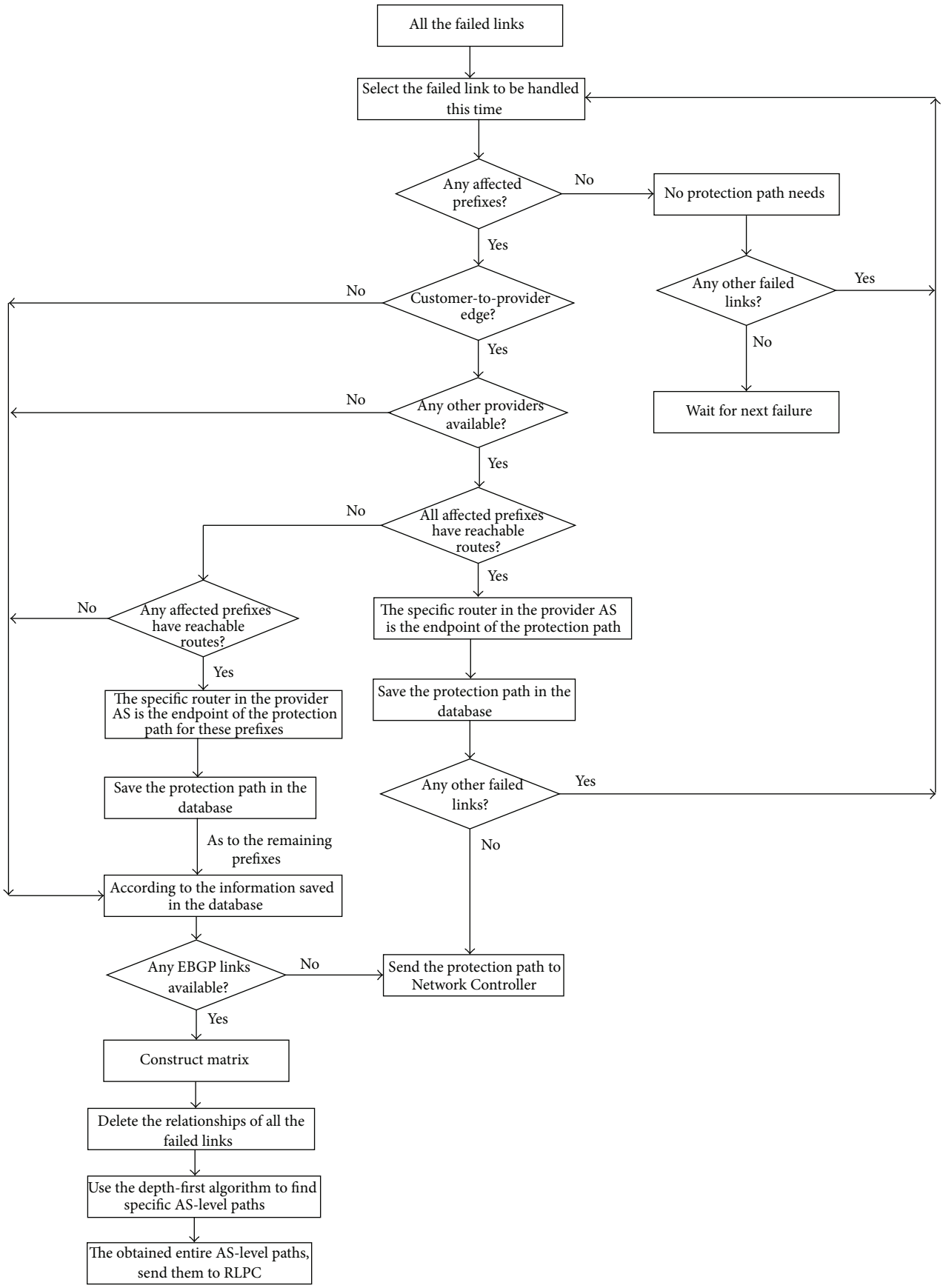


Figure 6: The flow chart of original ASPC.

which enables RLPC to perform the translation quickly and effectively. The model can be expressed as follows:

$$\begin{aligned}&\text{CovtPath (AS\_Paths, AS\_Info)}\\&\quad\{\text{Router\_level\_Paths} \mid \text{AS\_Paths} \subseteq \text{AS\_PATHS}\\&\qquad\&\&\text{AS\_Info} \subseteq \text{DB}_{\text{AS\_info}}\},\end{aligned} \tag{7}$$

where AS_Paths is the AS-level paths to be converted, Router_level_Paths is the set of refined router-level paths, AS_PATHS is the set of AS-level paths obtained from ASPC, AS_Info is the information of ASes involving the path conversion, and $\text{DB}_{\text{AS\_info}}$ is the information of ASes saved in the database.

(3) Routing Policy Filtering. Routing policy plays a crucial role in the interdomain routing; ASes can use routing policy to realize their special demands. For example, for security or privacy reasons, they may refuse to accept some prefixes or announce some prefixes, and they may also restrict some AS paths, which make the obtained AS-level protection paths unavailable for some prefixes. To make the obtained protection paths more accurate and reasonable in the interdomain routing system, BGP routing policies are considered to see whether the router-level paths comply with routing policies. The function of RPF is to filter out the paths which do not conform to the routing policies. The model can be expressed as follows:

$$\begin{aligned}&\text{PolicyFilter}\left(\text{Router\_level\_paths, routing\_policy}\right)\\&\quad\{\text{PolicyCompliant\_Paths} \mid \text{Router\_level\_paths}\\&\qquad\subseteq \text{Router\_Level\_Paths}\\&\qquad\&\&\text{routing\_policy} \subseteq \text{DB}_{\text{rp}}\},\end{aligned} \tag{8}$$

where PolicyCompliant_Paths is the router-level paths that conform to the routing policies, Router_level_paths is the router-level paths to be filtered by RPF, Router_Level_Paths is the set of router-level paths obtained from RLPC, routing_policy is the policy involving the policy filtering, and DB_{rp} is the routing policy saved in the database.

(4) Optimal Path Selection. After RPF filters the path using routing policies, for each affected prefix, OPS aims to find the optimal protection tunnel for them. Specially, OPS seeks to find the earlier router that terminates the whole protection tunnel; after the special router, the affected packets can be forwarded normally without travelling along the entire protection paths. For each of the sequential router sets corresponding to the routing compliant paths, OPS finds the earlier router with the help of BGP routing tables. If such a router is not found, it selects the best paths using the simplified BGP decision processes. The model can be expressed as follows:

$$\text{OPT\_PTS} = \begin{cases}\text{ProtectionTunnel}_{\text{Rt}}, & \text{if Rt is found},\\ \text{ProtectionTunnel}_{\text{dp}}, & \text{otherwise},\end{cases} \tag{9}$$

where OPT_PTS is the set of optimal protection paths, Rt is the found earlier router that terminates the protection tunnel, $\text{ProtectionTunnel}_{\text{Rt}}$ represents the protection tunnels where Rt is the endpoint router, and $\text{ProtectionTunnel}_{\text{dp}}$ is the obtained protection tunnels that use simplified BGP decision process.

It should be noted that the design of the optimal path selection has guaranteed our approach to achieve loop-free routing. There are two situations that should be considered. One is that the optimal protection path does exist; that is, OPS finds the earlier special router that can shorten the length of the protection tunnel, so the affected packets can be forwarded along the shortened protection tunnel. After that, they are forwarded to the destination according to the BGP routing tables. The other one is that the earlier special router does not exist; that is, the protection tunnel is an entire protection path. As the protection path is a sequence of ASes between the AS associated with the failed links and the AS owing the affected prefix, the affected packets can be forwarded to their destinations along the entire AS-level protection paths. As described above, the affected packets can be forwarded to their destinations along an entire path under the two situations, which can achieve the loop-free routing.

3.3. Deactivation of Protection Path. To accurately deactivate the protection tunnels under multiple link failure scenarios, it should consider the special condition that the constructed protection tunnels or part of them may be shared by multiple link failures. In this case, it must make sure that the protection tunnel is useless before it is deactivated. To this end, we adopt a number to represent the times a protection tunnel was shared. The protection tunnel can be removed only when the value of the number is equal to 0. The model can be expressed as follows:

$$\text{Deactivation\_PT} = \begin{cases}\text{true}, & \text{if } n = 0\&\&\text{pr=true},\\ \text{false}, & \text{otherwise},\end{cases} \tag{10}$$

where Deactivation_PT denotes whether to deactivate the protection tunnel, n is the times the protection tunnel was shared, and pr denotes whether the path relevant to special prefix is reachable.

4. Experiments and Results Analysis

In this section, we evaluate the performance of SDFRR-ASR on the real-life topology as shown in Figure 7. There are 26 routers in 13 distinct ASes, where routers are simulated by computers running CentOS 6.4 and Quagga 0.99.22. Moreover, OpenFlow 1.0 is installed to support OpenFlow, and Zebra and BGPd programs are also well configured. Each router represents a BGP router; the AS relationships between interconnected AS pairs are presented in Figure 7. The experimental platform is an x86 based server with Intel (R) Xeon (R) CPU E5645 @2.40 GHz and 96 GB RAM running VMware ESXi 5.1.0. Our algorithms were coded in Java on the platform of Eclipse and implemented on an x64 based desktop computer with Intel (R) Core

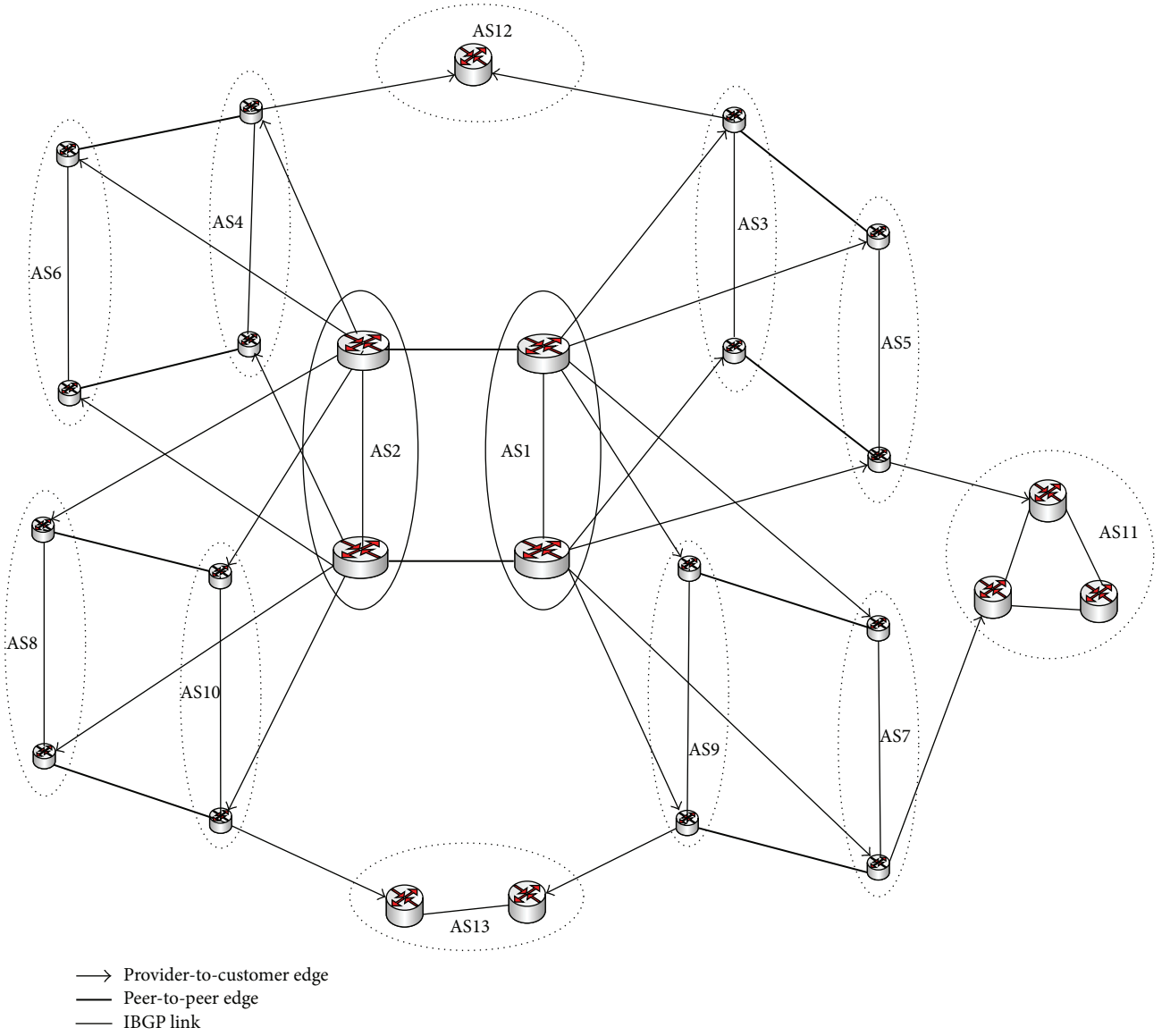


Figure 7: Experimental topology.

(TM) i3-3220T and 2.80 GHz CPU with 4 GB RAM under the Microsoft Windows 8.1 Professional operating system. Network Controller is deployed on an independent server; it is connected to each router by control links to ensure the safety and reliability of the connection. It should be noted that the function of the control link is implemented by network management networks in practice.

4.1. Experimental Metrics. Several metrics are used in our experiments to evaluate the performance of SDFRR-ASR, and the details are described as follows.

(1) Failure Recovery Time. The failure recovery time indicates the speed of recovery upon the occurrence of failures. In our approach, we mainly focus on concurrent failure scenarios and continuous failure scenarios. With regard to the concurrent failure scenario, the failure recovery time is the difference between the time of the last constructed protection tunnel and the time of the first happening failure, which is calculated as follows:

$$T_{\text{concur}} = T_{\text{lc}} - T_{\text{fd}}, \tag{11}$$

where T_{lc} is the time for the last protection tunnel construction and T_{fd} is the time for the first failure detection.

As for the continuous failure scenario, the failure recovery time is the sum of all failure's recovery times. The formula can be represented as follows:

$$T_{\text{continu}} = T_{\text{fr}} + T_{\text{sr}} + \cdots + T_{nr}, \tag{12}$$

where T_{fr} is the first failure recovery time, T_{sr} is the second failure recovery time, and T_{nr} is the nth failure recovery time. The failure recovery time $T_{continu}$ is the sum of these parts.

(2) Packet Loss Rate. Generally, network reacts to failures by starting BGP convergence; the packets affected by the failed link will be dropped during this period. In this paper, we use Spirent TestCenter to send data flows on the failed links, and packet loss rate (PLR) is calculated according to the following formula:

$$\mathrm{PLR} = \frac{N_{\mathrm{drop}}}{N_{\mathrm{sum}}} = \frac{N_{\mathrm{drop}}}{v \times t/s}, \tag{13}$$

where N_{drop} is the number of packets dropped during the period, N_{sum} is the sum of packets, v is the speed of the data flow sent by TestCenter, t is the time during the period, and s is the size of each packet send by TestCenter.

(3) Control Overhead. The control overhead is introduced by Network Controller sending commands to routers; it includes two parts: one is that sending commands to routers to construct protection tunnels and the other one is that sending commands to routers to withdraw flow entries.

When Network Controller sends commands to install OpenFlow flow entries, we assume that the hop from the Network Controller to the router on the protection path is h_i and the size of control message is bt_i byte, so the control overhead introduced by the Network Controller is calculated as the following equation, where n is the number of routers on the protection path:

$$C_{\mathrm{install}} = \sum_{i=1}^{n} \mathrm{bt}_i \times h_i. \tag{14}$$

When Network Controller sends commands to remove OpenFlow flow entries, we assume that the hop from the Network Controller to the router is h_j and the size of withdraw message is bt_j byte; the withdraw control overhead introduced by Network Controller is calculated as the following equation, where m is the number of routers on the protection path:

$$C_{\mathrm{withdraw}} = \sum_{j=1}^{m} \mathrm{bt}_j \times h_j. \tag{15}$$

All in all, the total control overhead is the sum of $C_{install}$ and $C_{withdraw}$; the formula is described as follows:

$$C_{\mathrm{total}} = C_{\mathrm{install}} + C_{\mathrm{withdraw}} = \sum_{i=1}^{n} \mathrm{bt}_i \times h_i + \sum_{j=1}^{m} \mathrm{bt}_j \times h_j. \tag{16}$$

4.2. Experimental Results. As we are interested in concurrent failure scenarios and continuous failure scenarios, two sets of experiments with AS relationships were conducted to evaluate the performance of our approach. Due to space limitations, only up to four failures are shown in the results, but it is important to note that our approach can handle multiple link failures, rather than just four.

4.2.1. Failure Recovery Time. We first conduct experiments to compare the failure recovery time between SDFRR-ASR and normal BGP protocol; that is, the MRAI timer for eBGP sessions is set to 30 seconds and that for iBGP sessions is set to 5 seconds. The failure recovery time in SDFRR-ASR is described in Section 4.1, while it is the convergence time of BGP protocol.

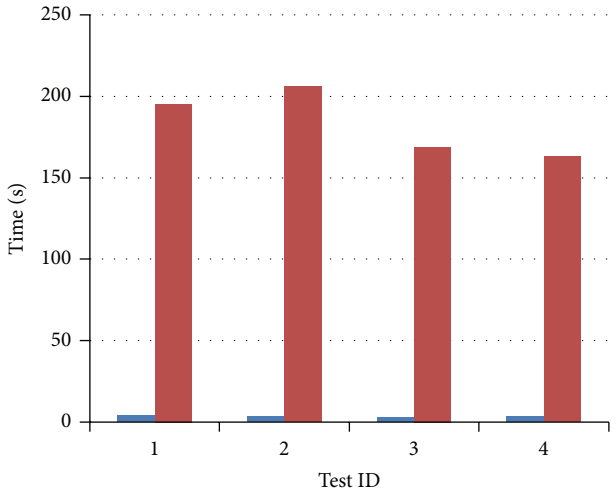
Figures 8 and 9 show the average failure recovery time under concurrent failure scenarios and continuous failure scenarios, respectively. We can see that SDFRR-ASR outperforms BGP protocol under all failure scenarios; it can provide faster failure recovery time than BGP protocol. The reason is that SDFRR-ASR takes AS relationships into account to provide protection for link failures, which enables it to quickly compute and construct protection paths. Another observation is that the average failure recovery time of SDFRR-ASR increases with the number of failures. The more failures occur, the longer recovery time is needed, so this metric of our approach is affected by the number of failures. Figures 8 and 9 show also that, compared to concurrent failure scenarios, the average failure recovery time under continuous failure scenarios is longer. That is because the computation for them is different; the former is from the first failure detected to the last protection tunnel constructed, while the latter is the sum of every failure's recovery times.

4.2.2. Packet Loss Rate. In order to compare the packet loss rate between SDFRR-ASR and normal BGP protocol, experiments were conducted under different failure scenarios. The size of the packets sent by Spirent TestCenter is 128 bytes.

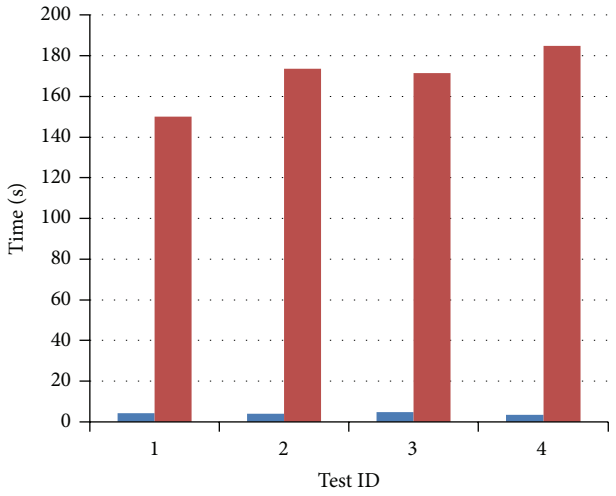
Figures 10 and 11 show the average packet loss rate under concurrent failure scenarios and continuous failure scenarios, respectively. It can be observed under all failure scenarios that SDFRR-ASR undergoes a very small packet loss rate; it provides significantly reduced packet loss rate compared to the BGP protocol. There are two reasons for which the loss of SDFRR-ASR in our experiments is nonzero; one is that it relies on the constructed protection tunnel to protect affected packets (some packets will be lost before protection tunnel works), and the other is that if the existing protection tunnel is affected by the subsequent failures under continuous failure scenarios, the affected tunnel should be deactivated and new protection path will be computed, which will further cause some packet loss. However, Figures 10 and 11 show that SDFRR-ASR has significantly lower packet loss rate than that of normal BGP protocol under all failure scenarios. This is because our approach can quickly react to the failures by providing protection paths, and the affected packets can be forwarded based on the flow tables after the protection path works, which can effectively reduce the packet loss rate.

4.2.3. Control Overhead. We investigate the control overhead incurred by our approach upon concurrent failure scenarios and continuous failure scenarios. The results are presented in Tables 1 and 2, respectively.

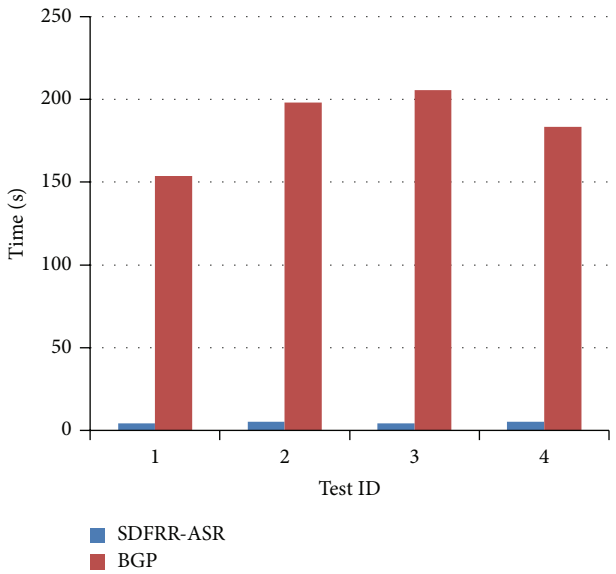
Table 1 details the control overhead incurred by SDFRR-ASR under concurrent failure scenarios, and Table 2 details the control overhead under continuous failure scenarios.



(a) Two concurrent failures

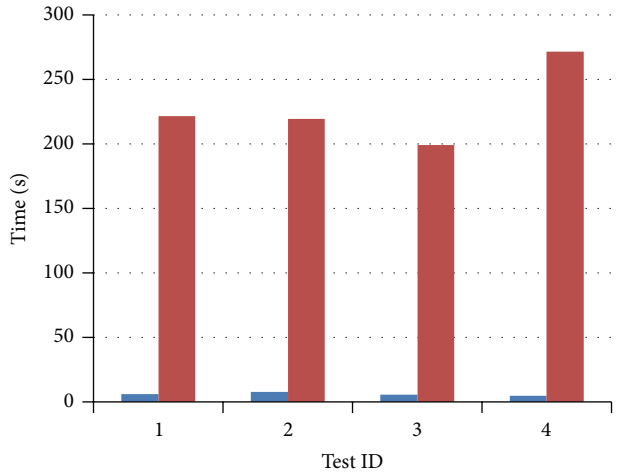


(b) Three concurrent failures

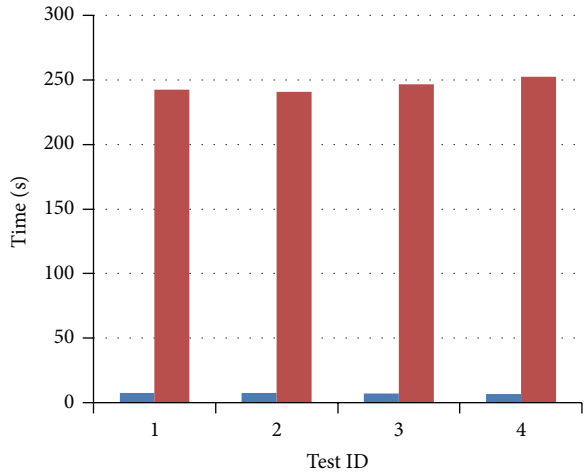


(c) Four concurrent failures

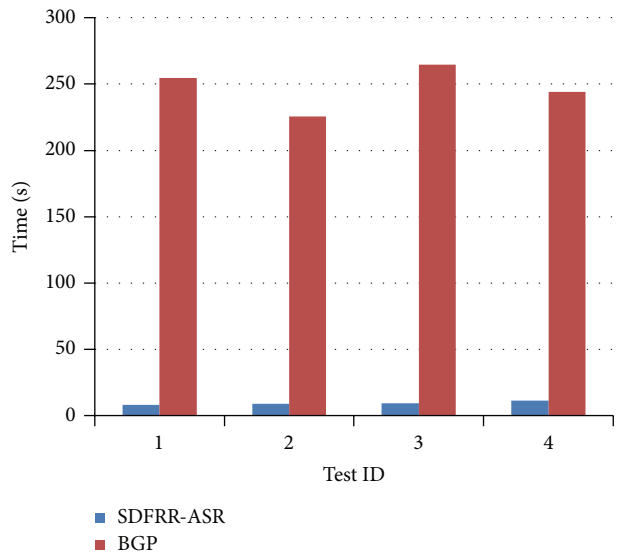
Figure 8: Failure recovery time under concurrent failures.



(a) Two continuous failures

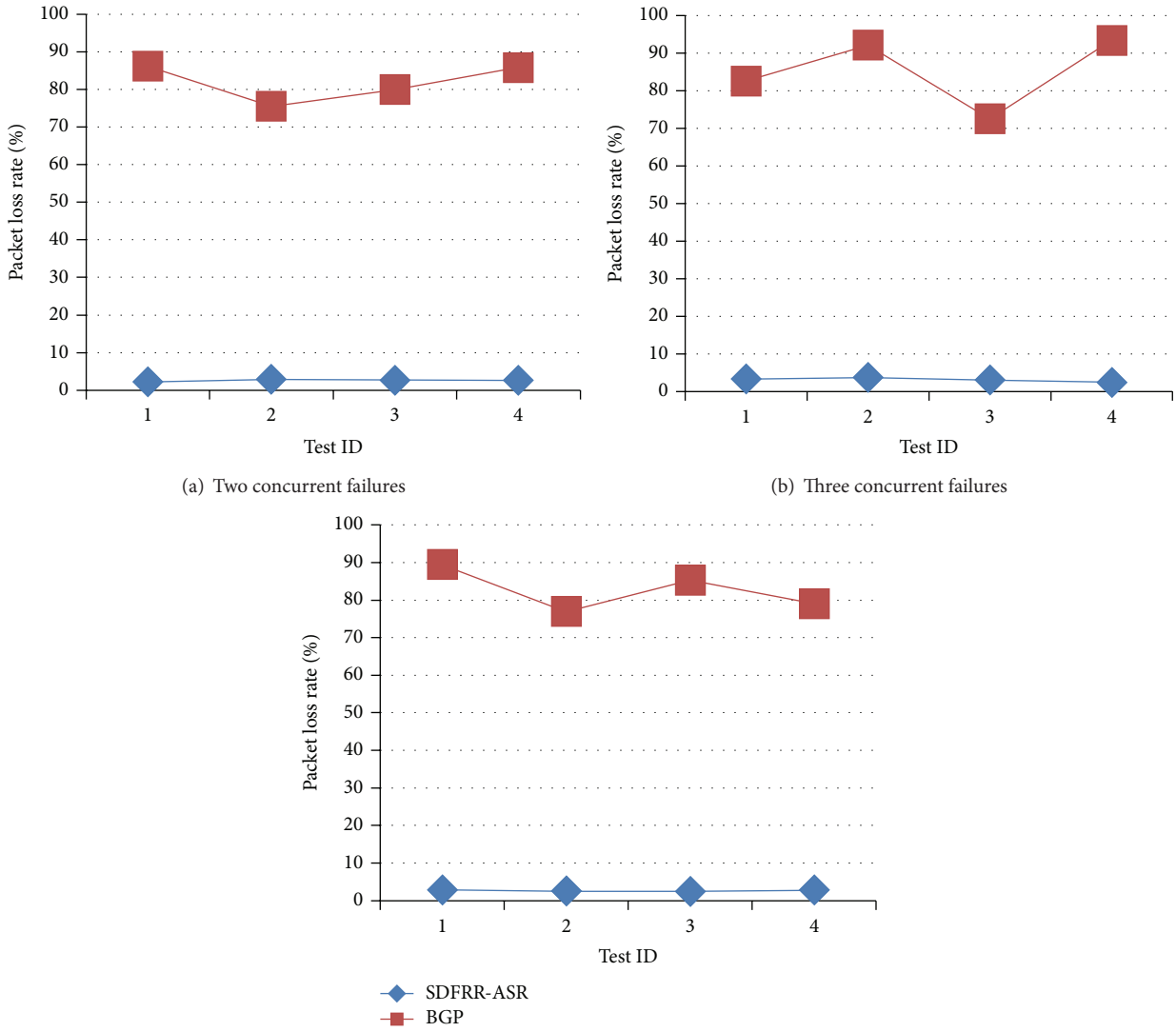


(b) Three continuous failures



(c) Four continuous failures

Figure 9: Failure recovery time under continuous failures.



(a) Two concurrent failures

(b) Three concurrent failures

(c) Four concurrent failures

Figure 10: Packet loss rate under concurrent failures.

We highlight three important aspects of the results. First, the control overhead increases with the number of failures, since SDFRR-ASR can provide protection for every failure by constructing protection tunnels. Second, the length of the protection path has a great influence on the control overhead; this is because Network Controller should send commands to relevant routers on the protection paths in order to construct protection tunnels. But compared with the overhead on the backbone network, the control overhead introduced by our approach is small, which is insignificant and will not exacerbate the network performance. Third, special cases in the continuous failure scenarios introduce larger control overhead than that of normal continuous failure scenarios. In every set of tests as shown in Table 2, the first two tests are normal continuous failures, while the following two tests are special cases. It can be observed that the control overhead introduced by the first two tests is less than the following two tests. The reason is that Network Controller should first send commands to deactivate the affected protection tunnels, and then it sends commands to construct new protection tunnels for the affected failed links, which will incur more control overhead.

5. Conclusions

Interdomain links play a critical role in the global Internet routing system, but their failures are common on the Internet, while BGP routers fail to react quickly to recover from them. Besides, AS relationships also have an impact on the interdomain routing. In this paper, we present software defined AS-level fast rerouting to handle multiple link failures in one administrative domain by considering AS relationships.

In the architecture of our scheme, the control plane is separated from the data plane as SDN. Our approach aims to provide AS-level fast rerouting with AS relationships

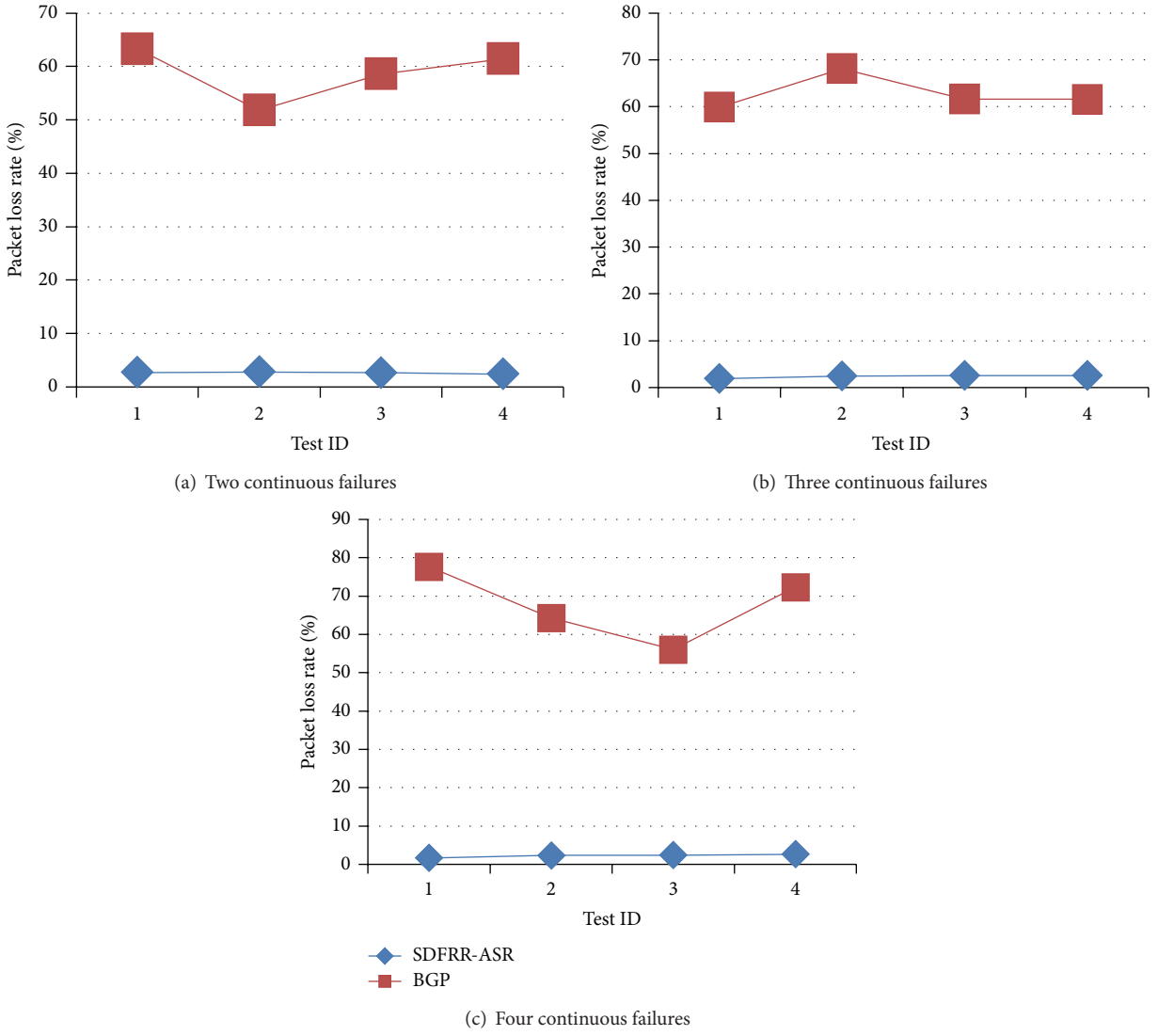


Figure 11: Packet loss rate under continuous failures.

Table 1: Control overhead incurred by concurrent failures.

Failure Scenario	Test ID	Establish tunnel (KB)	Remove tunnel (KB)	Total (KB)	Average (KB)
Two concurrent failures	1	0.9140	0.8164	1.7304	1.7255
	2	0.9101	0.8125	1.7226	
	3	0.9121	0.8144	1.7265	
	4	0.9101	0.8125	1.7226	
Three concurrent failures	1	1.3691	1.2226	2.5917	2.5887
	2	1.3671	1.2207	2.5878	
	3	1.3691	1.2226	2.5917	
	4	1.3652	1.2187	2.5839	
Four concurrent failures	1	1.8222	1.6269	3.4491	3.4452
	2	1.8203	1.6250	3.4453	
	3	1.8183	1.6230	3.4413	
	4	1.8203	1.6250	3.4453	

Table 2: Control overhead incurred by continuous failures.

Failure Scenario	Test ID	Establish tunnel (KB)	Remove tunnel (KB)	Total (KB)	Average (KB)
Two continuous failures	1	0.9121	0.8144	1.7265	2.1532
	2	0.9101	0.8125	1.7226	
	3	1.3652	1.2167	2.5819	
	4	1.3632	1.2187	2.5819	
Three continuous failures	1	1.3671	1.2207	2.5878	3.0140
	2	1.3652	1.2187	2.5839	
	3	1.8183	1.6230	3.4413	
	4	1.8183	1.6250	3.4433	
Four continuous failures	1	1.8203	1.6250	3.4453	3.8797
	2	1.8417	1.6269	3.4686	
	3	2.2753	2.0312	4.3065	
	4	2.2714	2.0273	4.2987	

under multiple link failures. Taking different failure scenarios into account, we design efficient algorithms to automatically and dynamically find policy-compliant protection paths by considering AS relationships, in addition to routing policies and BGP decision rules; then OpenFlow forwarding rules are installed on relevant routers associated with protection paths to provide fast packet rerouting. Experimental results and analysis demonstrate its feasibility and effectiveness. Our future research will involve how to establish protection tunnels if not all the ASes are SDN-capable. We will also consider the case when each AS belongs to a separate ISP.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The research presented in this paper is supported by the National Natural Science Foundation of China, Distinguished Young Scholar (61425012), and Fundamental Research Funds for the Central Universities (2014PTB-00-02).

References

[1] G. Iannaccone, C.-N. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot, "Analysis of link failures in an IP backbone," in *Proceedings of the 2nd ACM SIGCOMM Internet Measurement Workshop (IMW '02)*, pp. 237–242, Marseille, France, November 2002.

[2] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, and C. Diot, "Characterization of failures in an IP backbone," in *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies and Conference on Computer Communications (IEEE INFOCOM '04)*, pp. 2307–2317, Hong Kong, March 2004.

[3] Q. Shen, B. Jiang, and V. Cocquempot, "Adaptive fuzzy observer-based active fault-tolerant dynamic surface control for a class of nonlinear systems with actuator faults," *IEEE Transactions on Fuzzy Systems*, vol. 22, no. 2, pp. 338–349, 2014.

[4] Y. Rekhter, T. Li, and S. Hares, "A border gateway protocol 4 (BGP-4)," Tech. Rep. RFC 4271, 2006.

[5] Q. Li, M. Xu, J. Wu et al., "A unified approach to routing protection in IP networks," *IEEE Transactions on Network and Service Management*, vol. 9, no. 3, pp. 306–319, 2012.

[6] Y. Afek, A. Bremler-Barr, and S. Schwarz, "Improved BGP convergence via ghost flushing," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 10, pp. 1933–1948, 2004.

[7] D. Pei, X. Zhao, L. Wang et al., "Improving BGP convergence through consistency assertions," in *Proceedings of the IEEE 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '02)*, vol. 2, pp. 902–911, New York, NY, USA, June 2002.

[8] D. Pei, M. Azuma, D. Massey, and L. Zhang, "BGP-RCN: improving BGP convergence through root cause notification," *Computer Networks*, vol. 48, no. 2, pp. 175–194, 2005.

[9] N. Kushman, S. Kandula, D. Katabi, and B. M. Maggs, "R-BGP: staying connected in a connected world," in *Proceedings of the 4th USENIX Symposium on Networked Systems Design & Implementation (NSDI '07)*, pp. 341–354, April 2007.

[10] F. Wang and L. Gao, "Path diversity aware interdomain routing," in *Proceedings of the Conference on Computer Communications (IEEE INFOCOM '09)*, pp. 307–315, Rio de Janeiro, Brazil, April 2009.

[11] O. Bonaventure, C. Filsfils, and P. Francois, "Achieving sub-50 milliseconds recovery upon BGP peering link failures," *IEEE/ACM Transactions on Networking*, vol. 15, no. 5, pp. 1123–1135, 2007.

[12] L. Gao, T. G. Griffin, and J. Rexford, "Inherently safe backup routing with BGP," in *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '01)*, vol. 1, pp. 547–556, Anchorage, Alaska, USA, April 2001.

[13] W. Xu and J. Rexford, "MIRO: multi-path interdomain routing," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '06)*, pp. 171–182, ACM, Pisa, Italy, September 2006.

[14] S. 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*, October 2010.

[15] 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.

[16] B. Yang, J. Liu, S. Shenker, J. Li, and K. Zheng, "Keep forwarding: towards k-link failure resilient routing," in *Proceedings of IEEE INFOCOM*, pp. 1617–1625, Toronto, Canada, April 2014.

[17] S. Kini, S. Ramasubramanian, A. Kvalbein, and A. F. Hansen, "Fast recovery from dual-link or single-node failures in IP networks using tunneling," *IEEE/ACM Transactions on Networking*, vol. 18, no. 6, pp. 1988–1999, 2010.

[18] A. Gopalan and S. Ramasubramanian, "Multipath routing and dual link failure recovery in IP networks using three link-independent trees," in *Proceedings of the 5th IEEE International Conference on Advanced Networks and Telecommunication Systems (ANTS '11)*, Bengaluru, Indian, December 2011.

[19] T. Elhourani, A. Gopalan, and S. Ramasubramanian, "IP fast rerouting for multi-link failures," in *Proceedings of the IEEE Conference on Computer Communications (IEEE INFOCOM '14)*, pp. 2148–2156, Toronto, Canada, April 2014.

[20] N. McKeown, T. Anderson, H. Balakrishnan et al., "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[21] Y. Yang, S. Chen, X. Li, and Y. Wang, "A framework of using OpenFlow to handle transient link failure," in *Proceedings of the International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE '11)*, pp. 2050–2053, Changchun, China, December 2011.

[22] C.-X. Li, X. Li, K. Li, H. Zhang, Y.-L. Shi, and S.-Z. Chen, "Toward software defined AS-level fast rerouting," *The Journal of China Universities of Posts and Telecommunications*, vol. 21, no. 6, pp. 100–108, 2014.

[23] J. Xia and L. Gao, "On the evaluation of AS relationship inferences," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '04)*, vol. 3, pp. 1373–1377, November-December 2004.

[24] L. Gao, "On inferring autonomous system relationships in the internet," *IEEE/ACM Transactions on Networking*, vol. 9, no. 6, pp. 733–745, 2001.

[25] S. Y. Qiu, P. D. McDaniel, and F. Monrose, "Toward valley-free inter-domain routing," in *Proceedings of the IEEE International Conference on Communications (ICC '07)*, pp. 2009–2016, Glasgow, UK, June 2007.