

## Proposed enhanced link failure rerouting mechanism for software-defined exchange point

Abdijalil Abdullahi<sup>1,2</sup>, Selvakumar Manickam<sup>1</sup>, Shankar Karuppayah<sup>1</sup>, Mahmood A. Al-Shareeda<sup>1</sup>

<sup>1</sup>National Advanced IPv6 Centre, Universiti Sains Malaysia, Penang, Malaysia

<sup>2</sup>Faculty of Computing, SIMAD University, Mogadishu, Somalia

### Article Info

#### Article history:

Received Dec 28, 2022

Revised Mar 23, 2023

Accepted Mar 24, 2023

#### Keywords:

Internet exchange point  
Link failure recovery  
Path computation  
Software defined exchange point  
Software defined network

### ABSTRACT

Internet eXchange point (IXP) is a way to optimize network bandwidth. It enables a platform in the different providers like internet service providers (ISPs) and content delivery providers (CDNs) to share their traffic through a common point. A software defined exchange point (SDX) is an IXP comprising of a programmable deployed software defined network (SDN) switching fabric to enhance the management of their services, but they met the performance issues. The previous studies proposed various mechanisms and frameworks that tackled with these issues, but they don't overcome overall challenges like link failure recovery for multi-hop-based SDX particularly packet processing delay and switch memory overhead. To cope with these issues, this paper proposed an enhanced link failure rerouting (ELFR) mechanism for multi-hop-based SDX. The objective of the proposed ELFR mechanism is to reduce the delay of packet processing to recover the link failure quickly and improve the path computation while ensuring the switch storage overhead. The present paper is an effort to retrospect and analyze the critical review of existing work for SDX. Furthermore, this paper provides a background of SDX, its components and its applications. Finally, this paper presents and compares the expected results of ELFR mechanism and related work.

*This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.*



### Corresponding Author:

Selvakumar Manickam  
National Advanced IPv6 Centre, Universiti Sains Malaysia  
11800 USM, Penang, Malaysia  
Email: selva@usm.my

## 1. INTRODUCTION

Internet eXchange points (IXPs) are network infrastructures where all internet players can interconnect with each other to improve the quality of their services and reduce transmission costs [1]–[3]. Today, IXPs played an essential role in the development of an advanced Internet ecosystem across the globe [4]–[7]. Therefore, research quantified the following as classic IXP structural design constraints: restricted monitoring and visibility, layer-2 switching fabric, border gateway protocol (BGP) control plane, and manual policy enforcement [8]–[11]. In addition, other challenges include load balancing, loops, broadcast storms and lack of advanced policies and less flexibility of inter-domain routing. These limitations can be categorized into data plane and control plane in BGP [12]–[14].

In order to overcome the above challenges, the IXP providers adopt the software defined network (SDN) paradigm, which separates the data plane from the control plane and provides advanced policies and powerful functionalities to manage their services and offer better services to their customers [14]–[17]. A new idea is known as software defined exchange point has emerged as a result of the deployment of software defined networks on internet exchange points called SDX. To give network operators better granularity and control

over forwarding, SDX has been designed as a model. Operators can develop at far shorter time scales than others and exchange traffic much more effectively using SDX than they can with current methods [18]–[20].

Figure 1 illustrates the architecture of SDN-enabled IXP. The SDX design is made up of four main parts: programmable fabric, route server, SDX controller, and applications. First, the programmable fabric is an OpenFlow switch which provides more flexibility than a traditional layer-2 switch. Second, the SDX controller monitors network activity and implements low-level rules created from high-level policies on the IXP fabric. To regulate the IXP fabric, applications create high-level policies. The SDX takes use of APIs to offer sophisticated management capabilities to IXP members. The route server also makes it possible for IXP members to exchange BGP routes. Two autonomous systems (ASes)—AS A and AS B are connected to the programmable IXP fabric to exchange their traffic.

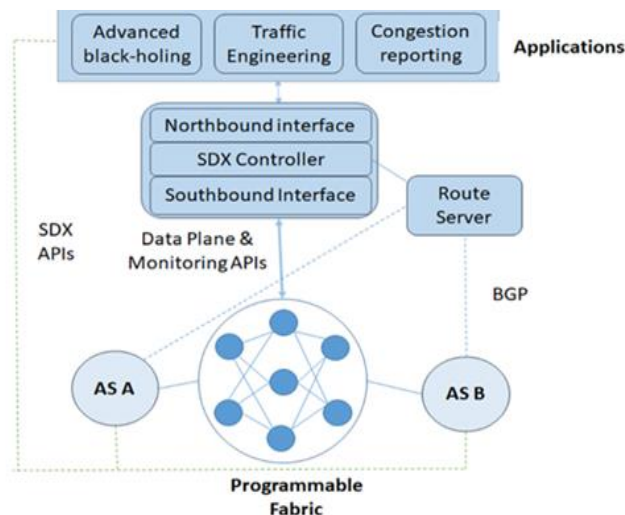


Figure 1. Architecture of SDN-Enabled IXP

The inter-domain problems faced by IXP operators were greatly reduced by SDX. Similarly, the previous studies suggested many approaches and frameworks tackle the issues related to the performance and scalability of software-defined exchange points (SDX). However, In the SDX environment, there are still outstanding problems and difficulties, such as link failure recovery and packet processing delays, which require highly advanced techniques to improve the quality of multi-hop IXP providers. The purpose of study is to address the challenges and issues of performance and scalability in the SDX environment and to propose an enhanced mechanism of link failure rerouting for multi-hop IXP based on SDN. This paper's contribution can be summed up as:

- Provide a review of the current frameworks and approaches for link failure recovery by categorizing the SDX-based and fast rerouting (FRR)-based frameworks.
- Investigate deeply the previous mechanisms and techniques by analyzing their weaknesses and strengths to realize the current challenges and limitations of link failure recovery.
- Propose enhanced link failure rerouting (ELFR); an enhanced mechanism to reduce delay of link failure rerouting in multi-hop SDX topology.
- Suggest an advanced packet processing algorithm to minimize the recovery time of link failure by exploiting the programming protocol-independent packet processors (P4) functionalities.
- Propose an improved path computation algorithm to compute the backup path recovery quickly upon link failure by exploiting the proactive link failure recovery mechanism of SDN.
- Present the expected result of proposed mechanism (ELFR) by measuring the metrics of recovery time of packet processing and calculating time of recovery path computation while considering cost switch memory.

The rest sections of this paper are structured as; section 2 reviews the related work for multi-hop-based SDX. Section 3 explains the background of this paper. Section 4 presents the design of the proposed ELFR mechanism. Section 5 evaluates and compares the expected result of the proposed ELFR. Finally, the conclusion and future work of this research are presented in section 6.

## 2. RELATED WORK

The current approaches and frameworks for recovering from link failure are presented in this section. The frameworks and approaches are categorized into two parts: SDX-based frameworks and FRR-based frameworks. SDX-based are frameworks which deployed on IXP providers by exploiting the SDN functionalities particularly multi-hop topology to improve link failure rerouting management. Similarly, the FRR-based are frameworks and approaches which are not yet deployed on multi-hop topology of SDX environment but they suggested strong approaches which can enable to reroute the failed packet quickly by using non-SDN mechanisms.

### 2.1. SDX-based frameworks

The IXP providers have met difficulties in managing the link failure rerouting especially multi-hop topologies. This section discusses two frameworks (ENDEAVOUR and Umbrella) that tackle the challenges of IXP fast failover recovery and proposed approaches. ENDEAVOUR [21] is an SDN framework for IXP operators and deployed on multi-hop IXP topologies. It reduces switching fabric issues related to broadcast traffic and expands the scalability of IXP providers. Additionally, this framework proposed three approaches to recover the link failure of multi-hop IXP topologies. These approaches are implementing duplicate outgoing policies, sending packets back to the ingress switch, and injecting recovery information into packets. These approaches are solved many challenges related to link failure recovery, but they are not overcome the delay of packet processing and switch memory overhead.

Umbrella [22] is an SDX model that can enable a strong and robust of switching fabric that minimizes the risks of controller dependability. In addition, Umbrella can be deployed on any single-hop or multi-hop IXP topology. For the purpose of dealing with data plane link failure, Umbrella relies on OpenFlow (OF) features. To respond to a failed link, OpenFlow essentially exploits the group fast failover feature. This feature allows monitoring of interfaces, switch port status, and forwarding action without the use of a controller. The data plane recovery failure is more challenging. Considering, the protocols (for example, bidirectional forwarding detection (BFD) and local link discovery (LLD)) are implemented by the Umbrella controller. Only the configuration of the edge switch with the backup route is changed by the Umbrella controller when a data plane link failure is discovered. Due to the dependability of OpenFlow features to reroute the failed packet quickly, the Umbrella framework was unable to overcome the limitations of link failure in multi-hop topology.

### 2.2. FRR-based frameworks

This section presents the general frameworks for FRR of link failure, including SWIFT, PURR, and FRR. SWIFT [23] offers a fast-reroute framework that enables routers to recover from remote failures in a short amount of time. SWIFT enables two techniques; First, SWIFT forecasts the remote failure based on a few BGP updates that were received. Second, SWIFT proposed an encoding scheme data plane that can quickly and flexibly update the affected forwarding entries. SWIFT reduced convergence time and solved the challenges of remote outages in transit networks. This framework tackled some challenges of fast rerouting but have met the limitations, which include rerouting unaffected prefixes that might result in overhead and affect the accuracy. Also, SWIFT was unable to reroute the large amount of prefixes, and it can only reroute the limited prefixes. The main issue that SWIFT ran into was the fact that it could take minutes before the first BGP update occurred after the corresponding data plane failure.

PURR [24] enables low failover latency and high switch throughput by avoiding packet recirculation. Similarly, PURR has a strong resilience for multiple concurrent failures, and also it enables recovery from the failure with minimal memory requirements. This mechanism presented an FRR primitive for programmable data planes, which allows for the implementation of existing failover mechanisms without recirculating packets and hence low failover latency and high switch throughput. The PURR was unable to implement in the SDX environment, which might improve the proposed mechanism of this paper.

FRR [25] proposed a fast rerouting framework to recover the router interruption and reroutes suddenly from failure. The pre-computed routing path was updated by this framework, enabling quick rerouting. This framework provides separate failover paths to reduce the number of paths impacted by interruption and packet loss rate for enhancing fast rerouting protection performance. This mechanism was applicable to the SDX environment, particularly on single topologies but not yet deployed on multi-hop topologies which may solve the challenges of link failure recovery. In addition, the mechanism makes the data plane updates by using two-stage forwarding rules, which might result in a delay in rerouting the interruption.

### 2.3. Critical review

The existing frameworks are critically reviewed in detail in this section. Today's IXPs perform a fast rerouting using legacy routing protocols such as open shortest path first (OSPF) and multi-protocol label switching (MPLS). The mechanisms reroute packets based on their selected egress ports and pre-calculated alternate routing when an internal link failure occurs. In single switch topology, normally rerouting packets is

a simple task in case link failure is raised. In multi-hop IXPs topology, performing packet rerouting is more challenge when a link failure occurs because it affects different switches of IXPs [21]. IXPs also enable rerouting packets by exploiting OpenFlow mechanisms for failover recovery, but these mechanisms are not highly efficient for managing fast rerouting packets, and they also do not have great performance for packet processing [22].

Although many link failure rerouting mechanisms have been proposed such SWIFT [23], PURR [24] and FRR [25]. These mechanisms performed fast rerouting efficiently, but they are not implemented on multi-hop SDX topology. Therefore, these mechanisms can apply to multi-hop SDX topologies to minimize packet processing delays and also improve previous fast failover recovery mechanisms.

In summary, Table 1 summarizes the proposed mechanisms deployed, their strengths, and their weaknesses of the existing ENDEAVOUR [21] framework. These approaches solved some challenges, but still there are limitations like packet processing delay and switch memory overhead. In summary, Table 2 summarizes the mechanisms used, their strengths, and their weaknesses of the existing Umbrella [22] framework. These mechanisms reduce some limitations but unfortunately, they depend on the OpenFlow features that result the delay of link failure recovery. In summary, Table 3 summarizes the proposed mechanisms used, their strengths, and their weaknesses of the existing SWIFT [23] framework. These mechanisms tackled some challenges of fast rerouting, but they met the limitations, which include rerouting unaffected prefixes that might result in overhead and affect the accuracy. Additionally, the proposed mechanisms were not able to reroute the large amount of prefixes, and it can reroute the limited prefixes.

Table 1. Summary of existing ENDEAVOUR framework for link failure rerouting

Proposed approach	Strength	Weakness
Duplicating outbound policies	Replicating all inbound and outbound policies of members across all switches.	Extensive duplication of forwarding state.
Sending packets back to the ingress switch	Eliminates unwanted forwarding state duplication.	Increased packet latency and bandwidth waste.
Injecting recovery information into packets	Solved all overheads of the above approaches.	Delay in packet processing and switch memory costs.

Table 2. Summary of existing Umbrella framework for link failure rerouting

Proposed mechanism	Strength	Weakness
OpenFlow group fast failover	Independently and without a controller, monitors the state of the ports, interfaces, and switch forwarding operations.	Unable to recover the data plane failure without controller.
Bidirectional forwarding detection (BFD) protocol and Local Link Discovery Protocol (LLDP)	It solved the challenge of recovering data plane failure which above mechanism caused.	It depends on the controller that causes the recovery delay.

Table 3. Summary of existing SWIFT framework for link failure rerouting

Proposed mechanism	Strength	Weakness
SWIFT inference algorithm	Predicts which prefixes will be affected with low outage notification. Prefixes affected by the failure are being rerouted on routes that are not affected.	Unable to manage the speed and accuracy of rerouting the affected prefixes simultaneously because it might have rerouted unaffected prefixes.
SWIFT encoding scheme	Makes it possible to update the affected forwarding entries quickly and easily.	Able to reroute the limited prefixes and not rerouted the much amount of prefixes. After a data plane failure, the first BGP update can take minutes to propagate.

Table 4 summarizes the proposed mechanisms used, their strengths and their weaknesses of existing PURR [24] and FRR [25] frameworks. In PURR, this mechanism presented an FRR primitive for programmable data planes, which allows for implementation of existing failover mechanisms without recirculating packets and hence low failover latency and high switch throughput. This mechanism is not able to deploy in the SDX environment related to this paper. This mechanism might improve the techniques of this study.

Meanwhile, in FRR this mechanism can be applied in the SDX environment, particularly on single topologies but not yet deployed on multi-hop SDX topologies which can solve the challenges of link failure recovery. In addition, the mechanism makes the data plane updates by using two-stage forwarding rules, which might result in a delay in rerouting the interruption.

Table 4. Summary of existing PURR and FRR frameworks for link failure rerouting

Proposed Framework	Proposed mechanism	Strength	Weakness
PURR [24]	Fast rerouting primitive for programmable data plane.	Provides low failover latency and high switch throughput. It can tolerate single and multiple failures. Prevents recirculating packets	Limited to data plane failures. Unable to deploy on multi-hop IXP topologies.
FRR [25]	Fast rerouting (FRR)	Diagnoses routing interruption. Implements fast rerouting technique. Reduces recovery time of interruption	Enables to make data plane updates by using two-stage forwarding rules which might takes a time to reroute the interruption. Unable to deploy on multi-hop IXP topologies.

### 3. BACKGROUND OF THE STUDY

First, this section explains the idea of SDX briefly. Second, it demonstrates the concepts of IXP, BGP, and SDN concisely. Third, it presents the description of the components for SDX. Lastly, the application of SDX is explained.

#### 3.1. Software-defined exchange point

SDX have emerged as an approach to offer network providers more particular control over forwarding [26]. Similarly, the deployment of software-defined networks on internet exchange points allows for supporting a rich policy expression among different participants. SDX offers a promise of significant flexibility and efficiency of inter-domain traffic delivery on the internet [27]. An SDX is an IXP made up of a programmable SDN switching fabric connected with a BGP route server that allows IXP members to communicate via BGP and an SDN controller that allows IXP members to override the default BGP routing behavior with more precise SDN policies. IXP members express SDN policies on both their inbound and outbound traffic [28]. The description of IXP, BGP, and SDN is provided in next section.

#### 3.2. Internet exchange point

IXP uses the concept of internet peering to interconnect domain networks such as ISPs and enterprises. Typically, internet peering refers to a business relationship by which two enterprises jointly offer access to each other's users. Similarly, IXP is a physical network infrastructure that facilitates the interconnection between two or more independent autonomous systems and exchanges their internet traffic. This network infrastructure offers small providers an effective way to make different peering relationships with one another, whether connecting through a direct link or an IXP [26].

#### 3.3. Border gateway protocol

BGP, the standard inter-domain routing protocol, is used by IXP providers to establish a peering connection between various network participants [29]. The way to establish connectivity between two networks (members) is to create a direct BGP session between two of their border routers. Primarily, when two IXP member ASes needed to start traffic exchange through the IXP switching fabric, they had to create a bilateral (BL) BGP peering session at the IXP [30]. The border gateway protocol, or BGP, has a significant capacity for scalability and can advance Internet development [31].

#### 3.4. Software-defined networking

SDN is the pattern that separates the control plane and data plane. SDN creates a centralized environment that enables to handle of the control information separately by the controller, which does not interfere with the data plane traffic [32], [33] open network operating system (ONOS), OpenDayLight (ODL), and Ryu are some of the SDN controllers. The data plane services are offered by the SDN switches. The initial packet will be routed to the SDN controller once it reaches an SDN switch. Flow entries are written to SDN switches by the SDN controller according to rules that govern policies. The packets can adhere to the necessary policy to reach the desired location [34].

#### 3.5. SDX components

SDX is paradigm that combines the ideas of software-defined network and internet exchange point to give IXP participants greater administration and highly programmable capabilities that allow for more robust policies and higher-quality services. The SDX architecture comprised four main components, namely, programmable fabric, route server, SDX controller, and SDX applications. The description of these components is provided in next section.

### 3.5.1. Programmable fabric

The SDX provides highly programmable switch fabric which enables to handle the forwarding packets effectively and quickly. The traditional switches do not have the capability of letting network administrators to direct the packet forwarding. They do not separate controlling from forwarding which can result significant cost. The SDX supports OpenFlow switch which offers greater flexibility and substantially simplifies data plane forwarding compared to the traditional layer-2 switch.

### 3.5.2. Route server

The route server (RS) is a feature designed for IXP operators that offers an alternative to full eBGP mesh peering among the service providers who have a presence at IXP. For each service provider, the RS offers eBGP route reflection with customized policy support. However, this component is part of traditional IXP architecture allows for the collection of BGP route information. The SDX architecture continues to use a route server which allows IXP members to share BGP routes and also stores each member's receiving route information and transfers without altering to the other IXP members.

### 3.5.3. SDX controller

SDX controller is the core component of software-defined exchange point architecture and responsible for managing control plane which instructs the switches where to transmit packets. High-level policies must be converted into low-level rules that can be on the IXP fabric by using the SDX controller. The SDX controller installs forwarding rules on switches and keep track of network activity using the Southbound Interface.

### 3.5.4. SDX applications

This component is responsible for creating a variety of network applications such as load balancing, firewalls, traffic engineering and many more to meet the internet provider's requirements. The network administrators can shape traffic without having to touch individual switches in the network. SDX application is responsible for creating high-level policies. This component utilizes the northbound interface to communicate with the SDX controller and manage network operations.

## 3.6. Applications of SDX

### 3.6.1. Application-specific peering

Video services like Netflix, Amazon Prime, and YouTube use a lot of bandwidth and significantly increase the volume of web traffic. This kind of traffic cannot be distinguished from real-time traffic via BGP. Due to the necessity for a special method to handle high-bandwidth services, SDX offers application-specific peering, which enables AS to exchange traffic in accordance with the requirements of the individual applications.

### 3.6.2. Inbound traffic engineering

BGP can successfully manage outgoing traffic but is unable to regulate incoming traffic due to its reliance on destination internet protocol (IP) prefix routing and use of indirect strategies like the ASP path, communities, and selective advertisement that have an effect on the neighbors. IXP participants, however, must enhance the system for controlling incoming traffic from other participants. In order to directly manage and control incoming traffic, SDX provides a framework that enables AS to build forwarding rules and implement on switches at the IXP.

### 3.6.3. Redirection through middle-boxes

Networks can take advantage of a variety of middle-box services, including firewalls, network address translators, and load balancers. These tools and equipment are expensive in major ISPs and difficult to install everywhere due to geographic restrictions. Large internet service providers (ISPs) attempted to use a mechanism to route traffic to particular middle-boxes, however doing so is difficult when employing routing protocols. So, deploying one or more middle-boxes, SDX allows ISPs to direct traffic and reroute particular subsets.

### 3.6.4. Wide-area server load balancing

As servers are typically used to respond to requests from clients, content providers must employ domain name system (DNS) to spread these requests throughout a number of servers. The DNS servers managed and balanced client queries within a number of obstacles. By broadcasting anycast prefixes and altering the destination IP address to match with the chosen hosting location based on any attributes in the packet header, due to the fact that client requests move between server clusters, SDX enables content producers to balance the load.

**3.6.5. Advanced black-holing**

A technique by which an AS might request that its neighbors ignore packets with a specific IP prefix. Consequently, black-holing is unable to distinguish between malicious and legal packets that resulted to drop every packet. Therefore, to minimize these dangers, SDX gives operators an advanced black-holing system that allows them to manage and drop rules at the finest possible level and only discard undesired packets.

**4. DESIGN OF PROPOSED ENHANCED LINK FAILURE REROUTING MECHANISM**

This section proposed a new mechanism called enhanced link failure rerouting (ELFR) to address the challenges highlighted in the critical review section. ELFR mechanism improves the performance of software-defined exchange point, SDX particularly link failure recovery for multi-hop topologies by reducing the delay of packet processing and improving path computation. The following subsections discuss the requirements of this mechanism, architecture and modules of ELFR mechanism to achieve the objective of this paper.

**4.1. Requirement of ELFR mechanism**

The main requirements of the proposed ELFR mechanism are to achieve the main goal of this research and to validate the output of the proposed mechanism. ELFR has to achieve the following requirements:

- ELFR should be based on the SDX multi-hop topology by exploiting P4 features.
- ELFR must avoid packet recirculation that increases packet processing latency when a link failure occurs.
- ELFR should not compute the recovery backup path after link failure is detected which results a delay of recovery process.
- ELFR must not deal with any normal forwarding packet as a recovery packet.

**4.2. Architecture of ELFR mechanism**

The architecture of the proposed ELFR Mechanism is presented in this section. The major objective of this suggested mechanism is to shorten the time it takes for multi-hop SDN-based IXPs to recover from link failure. The proposed ELFR mechanism minimizes the packet processing delay of link failure and improves the path computation to find a recovery backup path in multi-hop IXP topology. The design of the proposed mechanism is comprised of two modules as shown in:

**4.2.1. Packet processing module**

This module aims to capture the packets and forward them to the destination effectively. This module enables to classify the normal packets from recovery packets. If the packet is normal, the module will process and forward normally. The affected packet needs a mechanism that can help it recover quickly and efficiently from failure when there is a link failure.

**4.2.2. Path computation module**

This module aims to compute the recovery backup path of link failure. The module provides the shortest path to recover quickly from failure and reroute packets with the new path. This module is only responsible for calculating the shortest recovery backup path and installing the rules into switches. Figure 2 depicts an overview of the proposed mechanism design.

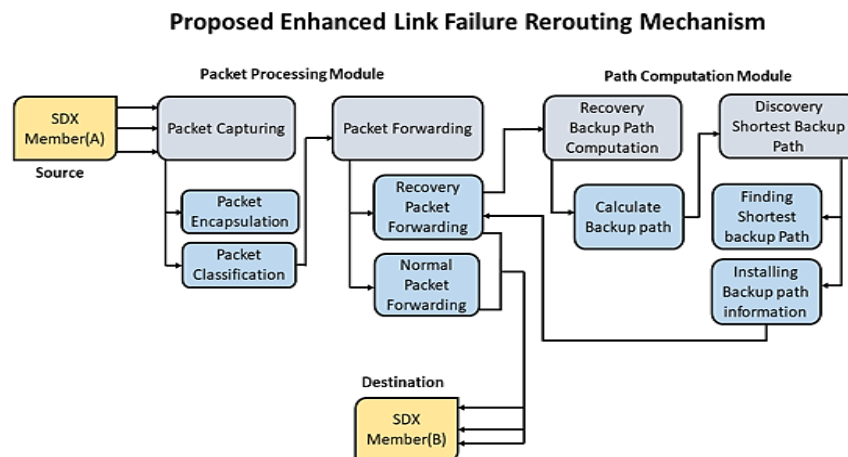


Figure 2. Architecture proposed ELFR mechanism for multi hop based SDX

### 4.3. Proposed ELFR mechanism

This section presents the details of proposed ELFR mechanism. ELFR mechanism contains two modules, namely packet processing module and path computation module. ELFR modules comprised various processes. These processes include packet capturing, packet forwarding for packet processing module, and recovery backup path computation and discovery shortest backup path for path computation module. Each process contains different steps such as packet encapsulation, packet classification, normal packet forwarding, recovery packet forwarding, calculate backup path, finding shortest backup path and installing backup path information. The main explanations of these modules are provided in next section.

#### 4.3.1. Packet processing module

This module aims to capture the packet by making two process of encapsulation and classification of packets. It also enables to forward the packets by separating which packet is normal and which packet needs a recovery. ENDEAVOUR and Umbrella proposed approaches by exploiting OpenFlow features that resulted in a packet processing delay and cost of switch memory [21], [22]. There are two link failure recovery techniques in the SDN architecture: proactive and reactive techniques. The proactive technique requires extra storage, which leads in switch memory cost, whereas the reactive mechanism adds latency to the failure recovery process. The ELFR mechanism adopts the proactive link failure recovery technique leveraging P4Neighbor features to decrease the delay of packet processing and minimize switch memory overhead [35]. The neighbor-based proactive link failure recovery approach described by P4Neighbor was not used in a multi-hop SDX environment. We employ P4Neighbor features in a SDX multi-hop topology to differentiate between normal and recovery packets in order to address the challenges raised in this work. The packet processing module consists of two processes: packet capturing and packet forwarding, as shown in Figure 3.

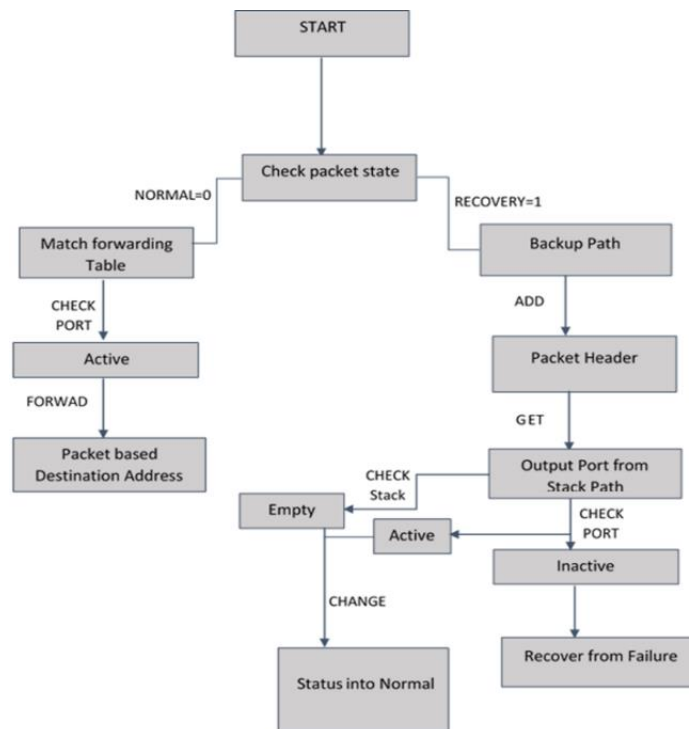


Figure 3. Workflow of packet processing module

- Packet capturing: this process contains two steps; packet encapsulation and packet classification. When a link fails, backup path information must be encapsulated into a packet header. To do so, this process uses P4 to capture and filter the packets from a network like autonomous systems (ASes). Packet classification enables to check the packet status and differentiate the packets, whether it is normal packets or recovery packets, to process packets effectively and quickly while preventing the forwarding of the unaffected packet as a recovery packet. This process distinguishes between normal and link failure recovery packets using a one-bit state field. The state 0 for normal packets and 1 for packets in link failure recovery. If a



packet is in the link failure recovery state, we must also include the backup path information in the header. Therefore, the role of this process is to capture and encapsulate the packet as well as distinguish the status of packets. When this process finishes the function, it will transfer to the next step to continue the process.

- Packet forwarding: this process comprises two steps; normal packet forwarding and recovery packet forwarding. This process handles the forwarding packets considering the previous step that differentiated the packets into two kinds. First, the normal packet is the packet which not effected the failure and needs to process normally. Second, a recovery packet is a packet that affects the failure and needs to reroute quickly from failure. The normal packet forwarding is responsible to forward the normal packets to the destination address. While the recovery packet forwarding is responsible to forward the recovery packets according to the backup path information in the packet header rather than normal packet. When the packet suffers a link failure in this situation, the switch inserts the backup path information into the custom packet header, and the following switches can complete the forwarding by retrieving the backup path information from the custom packet header.

**4.3.2. Path computation module**

This module aims to compute the recovery backup path before link failure occurs and discover the shortest backup path. The backup path must be installed in all switches by the controller in the traditional proactive failure recovery mechanism, which generates the backup path based on destination. This module implements a proactive link failure recovery mechanism using a neighbor-based backup path calculation approach as instead of the traditional destination-based backup path calculation method. This module contains two processes; recovery backup path computation and discovery shortest backup path.

- Recovery backup path computation: this process is responsible for calculating the backup path in advance before a link failure occur. In the SDN architecture, the controller determines the recovery backup path using both proactive and reactive mechanisms. When a link failure is discovered during reactive recovery, the mechanism computes the backup path after detecting the link failure that resulted delay in recovery process. Proactive recovery involves calculating the backup path in advance of a link failure, which adds storage overhead. Leveraging P4 capabilities, this step combines a neighbor-based backup path computation technique with a traditional proactive recovery mechanism. By adopting a neighbor-based backup path calculation, each switch can only store the backup paths of the other connected switches. The role of this process is to calculate the backup path only. After the calculation is finished, the next step will continue the process.
- Discovery shortest backup path: this process is responsible for finding the shortest backup path between switches and installing the backup path on switches. This process calculates the shortest path between any two witches. To get the shortest backup path, we need to find the path between each switch ( $S_x$ ) and its neighbor ( $S_y$ ) through the middle switch ( $S_m$ ). In order to achieve this, we navigate through all switches and save each path we find on  $S_m$  between two switch neighbors. The shortest backup path for the edge can be discovered after navigating all the switches. In this step, the shortest backup path that we have discovered is installed on each switch. Figure 4 shows the workflow of neighbor-based backup path calculation.

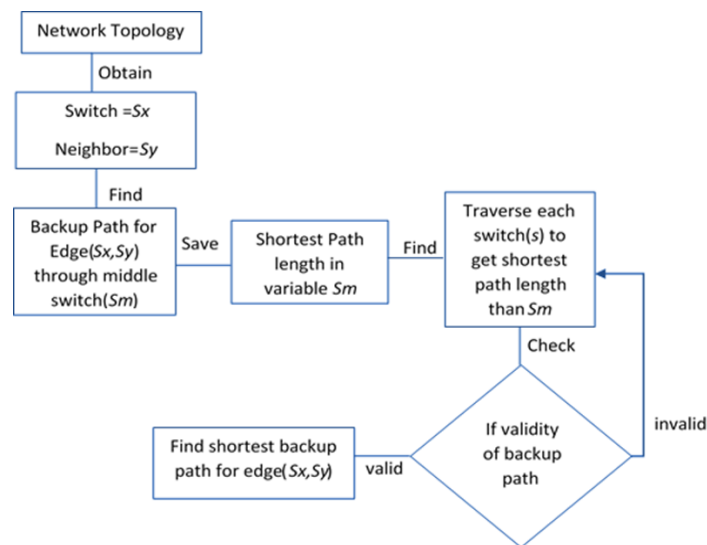


Figure 4. Flowchart of backup path calculation

## 5. EXPECTED RESULT

The proposed ELFR Mechanism compares with existing multi-hop SDX-based link failure recovery in terms of packet processing (recovery time), path computation (calculation time), and switch memory (overhead cost). Table 5 shows a comparison between the proposed ELFR mechanism and existing multi-hop SDX-based regarding link failure recovery metrics. This work measures the metrics by using “Low”, “Medium”, and “High”. So, the packet processing and path computation of ENDEAVOUR is “medium” because of there is a significant delay when processes the packet and also when computes the backup path in link failure recovery process. The switch memory of ENDEAVOUR is “High” because of it used reactive link failure recovery mechanism which requires much memory storage, so it needs to reduce this storage overhead. The all metrics of Umbrella are “High” because of there is a huge delay in terms of packet processing, path computation and switch memory. So, the Umbrella is not effective compared with the ENDEAVOUR. The ELFR is more effective than previous mechanisms. All metrics are “Low” in ELFR because the delay of packet processing and the backup path calculation are minimized and also ELFR is not required much storage of switch memory. The details are discussed the next sections.

Table 5. Comparison between the proposed ELFR mechanism and existing multi-hop SDX-based mechanisms

Metrics	ENDEAVOUR [21]	Umbrella [22]	ELFR
Packet processing	Medium	High	Low
Path computation	Medium	High	Low
Switch memory	High	High	Low

### 5.1. Packet processing

The previous frameworks proposed mechanisms that could enable to process of packets quickly when link failure is detected in a multi-hop-based SDX environment. These frameworks met the challenges of fast failover recovery and packet processing delay [21], [22]. However, the proposed mechanism, ELFR enables for the processing of the affected packet quickly and efficiently in multi-hop topologies by reducing the recovery time of link failure.

### 5.2. Path computation

The existing SDX-based frameworks in multi-hop topology are computed the recovery backup path calculation after link failure occurs that resulting in an operational delay to recover the failure quickly and depending on the OpenFlow features. So, the other frameworks proposed the mechanisms [24], [36] that can enable computing the recovery backup path before link failure occur, but they are not applied on multi-hop IXP topology. Therefore, the ELFR mechanism applies the technique that can quickly compute the path before link failure is detected by exploiting the existing mechanism to find the shortest path and reduce the calculation time of the backup path.

### 5.3. Switch memory

In the SDN environment, the controller calculates before link failure is detected and stores in ternary content-addressable memory (TCAM) switches that cause memory overhead. The existing frameworks [35], [36] proposed the mechanisms that reduce the switch memory overhead by using P4 features, but they are not implemented on multi-hop IXP topologies. So, the proposed ELFR mechanism attempts to reduce the memory overhead by exploiting the P4 features.

This study anticipates a different outcome from the past studies in terms of the parameters tested. The link failure recovery technique in particular benefits from ELFR's improved performance and scalability of SDX's multi-hop topology. The ELFR is a more scalable and efficient mechanism than the current SDX multi-hop topology mechanisms because it deploys P4's advanced features, which are better compared to OpenFlow's features in terms of processing packets and calculating backup paths in the event of link failure while minimizing the cost of switch memory.

## 6. CONCLUSION AND FUTURE WORK

SDX meets many performance and scalability challenges that cause operational management difficulties for IXP providers. These challenges include fast link failure recovery, which resulted in the delay of packet processing in the SDX environment. This paper proposed ELFR mechanism to improve the performance of multi-hop IXP topologies. The proposed ELFR reduces the delay of link failure recovery to

process packets quickly and improves the path computation to find the shortest backup path while minimizing the cost overhead of switch memory. In future work, instead of using multiple link failure recovery for backup path calculation, the proposed ELFR mechanism implements single failure recovery. Furthermore, the proposed ELFR mechanism uses a link-based failure scheme, which could result in a loop issue when numerous links fail and also reduce transmission efficiency.




## REFERENCES

- [1] T. Hoeschele, C. Dietzel, D. Kopp, F. H. P. Fitzek, and M. Reisslein, "Importance of internet exchange point (IXP) infrastructure for 5G: estimating the impact of 5G use cases," *Telecommunications Policy*, vol. 45, no. 3, 2021, doi: 10.1016/j.telpol.2020.102091.
- [2] L. Prehn, F. Lichtblau, C. Dietzel, and A. Feldmann, "Peering only? analyzing the reachability benefits of joining large IXPs today," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2022, vol. 13210 LNCS, pp. 338–366, doi: 10.1007/978-3-030-98785-5\_15.
- [3] Y. Dabone, T. F. Ouedraogo, and P. J. Kouraogo, "Improving the linkage of internet exchange points through connected ISPs ASes," in *Lecture Notes in Networks and Systems*, 2022, vol. 503 LNNS, pp. 180–187, doi: 10.1007/978-3-031-09073-8\_17.
- [4] D. Saraswat *et al.*, "Internet exchange points: a systematic case study in Indian context," in *Lecture Notes in Electrical Engineering*, vol. 875, Springer, 2022, pp. 731–745, doi: 10.1007/978-981-19-0284-0\_53.
- [5] S. Vakataki'Ofa, "Estimating the effects of internet exchange points on fixed-broadband speed and latency," *Asia-Pacific Sustainable Development Journal*, vol. 28, no. 2, pp. 39–68, 2022, doi: 10.18356/26178419-28-2-2.
- [6] V. Varadharajan, K. Karmakar, U. Tupakula, and M. Hitchens, "A policy-based security architecture for software-defined networks," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 897–912, 2018, doi: 10.1109/TIFS.2018.2868220.
- [7] M. Masoud, Y. Jaradat, and I. Jannoud, "A measurement study of internet exchange points (IXPs): History and future prediction," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 25, no. 1, pp. 376–389, 2017, doi: 10.3906/elk-1412-23.
- [8] D. Ó Briain, D. Denieffe, D. Okello, and Y. Kavanagh, "Enabling models of internet exchange points for developing contexts," *Development Engineering*, vol. 5, no. September 2020, p. 100057, 2020, doi: 10.1016/j.deveng.2020.100057.
- [9] H. Kumar, "Re-architecting internet exchange points for security and flexibility using software defined networking," Ph.D. dissertation, The School of Electrical Engineering and Telecommunications, The University of New South Wales, 2017, doi: 10.26190/unsworks/20248.
- [10] M. A. Al-Shareeda, S. Manickam, M. A. Saare, and N. C. Arjuman, "Proposed security mechanism for preventing fake router advertisement attack in IPv6 link-local network," *Indonesian Journal of Electrical Engineering and Computer Science (IJECS)*, vol. 29, no. 1, pp. 518–526, 2023, doi: 10.11591/ijeecs.v29.i1.pp518-526.
- [11] M. A. Al-Shareeda, S. Manickam, and M. A. Saare, "DDoS attacks detection using machine learning and deep learning techniques: analysis and comparison," *Bulletin of Electrical Engineering and Informatics (BEEI)*, vol. 12, no. 2, pp. 930–939, 2023, doi: 10.11591/eei.v12i2.4466.
- [12] L. F. C. Martins, I. Cunha, and D. Guedes, "An SDN-based framework for managing internet exchange points," *Proceedings - IEEE Symposium on Computers and Communications*, vol. 2018-June, pp. 996–1001, Nov. 2018, doi: 10.1109/ISCC.2018.8538695.
- [13] J. Ali and B. H. Roh, "An effective approach for controller placement in software-defined internet-of-things (SD-IoT)," *Sensors*, vol. 22, no. 8, p. 2992, 2022, doi: 10.3390/s22082992.
- [14] A. Abdullahi, S. Manickam, and S. Karuppayah, "A review of scalability issues in software-defined exchange point (SDX) approaches: state-of-the-art," *IEEE Access*, vol. 9, pp. 74499–74509, 2021, doi: 10.1109/ACCESS.2021.3069808.
- [15] H. Kumar, C. Russell, V. Sivaraman, and S. Banerjee, "A software-defined flexible inter-domain interconnect using ONOS," *Proceedings - European Workshop on Software-Defined Networks, EWSDN*, 2017, pp. 43–48, doi: 10.1109/EWSDN.2016.10.
- [16] T. E. Ali, Y. W. Chong, and S. Manickam, "Machine learning techniques to detect a DDoS attack in SDN: A systematic review," *Applied Sciences (Switzerland)*, vol. 13, no. 5, 2023, doi: 10.3390/app13053183.
- [17] T. E. Ali, Y. W. Chong, and S. Manickam, "Comparison of ML/DL approaches for detecting DDoS attacks in SDN," *Applied Sciences (Switzerland)*, vol. 13, no. 5, p. 3033, 2023, doi: 10.3390/app13053033.
- [18] R. Birkner, A. Gupta, N. Feamster, and L. Vanbever, "SDX-based flexibility or internet correctness? Pick two!," *SOSR 2017 - Proceedings of the 2017 Symposium on SDN Research*, 2017, pp. 1–7, doi: 10.1145/3050220.3050221.
- [19] J. Santos, T. Wauters, B. Volckaert, and F. De Turck, "Towards low-latency service delivery in a continuum of virtual resources: state-of-the-art and research directions," *IEEE Communications Surveys and Tutorials*, vol. 23, no. 4, pp. 2557–2589, 2021, doi: 10.1109/COMST.2021.3095358.
- [20] M. Chiesa *et al.*, "Inter-domain networking innovation on steroids: Empowering IXPs with SDN capabilities," *IEEE Communications Magazine*, vol. 54, no. 10, pp. 102–108, 2016, doi: 10.1109/MCOM.2016.7588277.
- [21] G. Antichi *et al.*, "ENDEAVOUR: A scalable SDN architecture for real-world IXPs," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2553–2562, 2017, doi: 10.1109/JSAC.2017.2760398.
- [22] M. Bruyere *et al.*, "Rethinking IXPs' architecture in the age of SDN," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 12, pp. 2667–2674, 2018, doi: 10.1109/JSAC.2018.2871294.
- [23] T. Holterbach, S. Vissicchio, A. Dainotti, and L. Vanbever, "Swift: Predictive fast reroute," *SIGCOMM 2017 - Proceedings of the 2017 Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 460–473, doi: 10.1145/3098822.3098856.
- [24] M. Chiesa *et al.*, "PURR: a primitive for reconfigurable fast reroute: hope for the best and program for the worst," *CoNEXT 2019 - Proceedings of the 15th International Conference on Emerging Networking Experiments and Technologies*, 2019, pp. 1–14, doi: 10.1145/3359989.3365410.
- [25] M. He, M. Zhang, X. Wang, J. Zhu, R. Peng, and Q. Wu, "A rerouting framework against routing interruption for secure network management," *IEEE Access*, vol. 7, pp. 143620–143630, 2019, doi: 10.1109/ACCESS.2019.2945777.
- [26] J. Griffioen, T. Wolf, and K. L. Calvert, "A coin-operated software-defined exchange," *2016 25th International Conference on Computer Communications and Networks, ICCCN 2016*, 2016, pp. 1–8, doi: 10.1109/ICCCN.2016.7568473.
- [27] J. Chung, J. Cox, R. Clark, and H. Owen, "FAS: Federated auditing for software-defined exchanges," in *Conference Proceedings - IEEE SOUTHEASTCON*, 2017, pp. 1–8, doi: 10.1109/SECON.2017.7925261.
- [28] A. Gupta *et al.*, "An industrial-scale software defined internet exchange point," in *Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2016*, 2016, pp. 1–14.
- [29] S. H. Warraich, Z. Aziz, H. Khurshid, R. Hameed, A. Saboor, and M. Awais, "SDN enabled and OpenFlow compatible network performance monitoring system," *arXiv:2005.07765*, pp. 1–10, 2020, doi: 10.48550/arXiv.2005.07765.




- [30] T. Böttger *et al.*, “Shaping the internet: 10 Years of IXP Growth,” *arXiv:1810.10963*, Oct. 2018, doi: 10.48550/arXiv.1810.10963.
- [31] W. J. A. Silva and D. F. H. Sadok, “A survey on efforts to evolve the control plane of inter-domain routing,” *Information (Switzerland)*, vol. 9, no. 5, 2018, doi: 10.3390/info9050125.
- [32] V. Muthumanikandan and C. Valliyammai, “Link failure recovery using shortest path fast rerouting technique in SDN,” *Wireless Personal Communications*, vol. 97, no. 2, pp. 2475–2495, 2017, doi: 10.1007/s11277-017-4618-0.
- [33] J. Ali, G. M. Lee, B. H. Roh, D. K. Ryu, and G. Park, “Software-defined networking approaches for link failure recovery: a survey,” *Sustainability (Switzerland)*, vol. 12, no. 10, 2020, doi: 10.3390/su12104255.
- [34] L. Barolli, M. Takizawa, F. Khafa, and T. Enokido, “Web, artificial intelligence and network applications”, *proceedings of the workshops of the 33rd International Conference on Advanced Information Networking and Applications (WAINA-2019)*, vol. 1150, 2020, doi: 10.1007/978-3-030-44038-1.
- [35] J. Xu, S. Xie, and J. Zhao, “P4Neighbor: efficient link failure recovery with programmable switches,” *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 388–401, 2021, doi: 10.1109/TNSM.2021.3050478.
- [36] Z. Li, Y. Hu, J. Wu, and J. Lu, “P4Resilience: Scalable resilience for multi-failure recovery in SDN with programmable data plane,” *Computer Networks*, vol. 208, 2022, doi: 10.1016/j.comnet.2022.108896.

## BIOGRAPHIES OF AUTHORS






**Abdijalil Abdullahi**    received a B.Sc. degree in information technology and an M.Sc. degree in networking and data communication from SIMAD University, Mogadishu, Somalia, in 2014 and 2018, respectively, where he is currently pursuing the Ph.D. degree with the National Advanced IPv6 Center (NAv6), Universiti Sains Malaysia (USM). His research interests include software-defined networking, inter-domain routing, and the internet ecosystem. He can be contacted at email: cabdijalil22@gmail.com.






**Selvakumar Manickam**    is currently working as an Associate Professor at National Advanced IPv6 Centre (NAv6), Universiti Sains Malaysia. His research interests include cybersecurity, internet of things, industry 4.0, and machine learning. He has authored and co-authored more than 160 articles in journals, conference proceedings, and book reviews and graduated 13 Ph.Ds. He has 10 years of industrial experience prior to joining academia. He is a member of technical forums at national and international levels. He also has experience building IoT, embedded, server, mobile, and web-based applications. He can be contacted at email: selva@usm.my.



**Shankar Karuppayah**    is currently a Senior Lecturer and researcher at National Advanced IPv6 Centre (NAv6), Universiti Sains Malaysia. He obtained his B.Sc. Computer Science (USM), Malaysia and the M.Sc. Software Systems Engineering (KMUTNB), Thailand. He obtained his Ph.D. in 2016 from Technische Universität Darmstadt in the field of Cyber Security. His main research interests are P2P Botnets, distributed systems and cyber security in general. To date, he has authored and co-authored many articles in journals, workshops, and conference proceedings. He is also a reviewer in many esteemed network and security journals. He can be contacted at email: kshankar@usm.my.



**Mahmood A. Al-Shareeda**    obtained his Ph.D. in Advanced Computer Network from University Sains Malaysia (USM). He is currently a researcher at National Advanced IPv6 Centre (NAv6), Universiti Sains Malaysia. His current research interests include network monitoring, internet of things (IoT), vehicular Ad hoc network (VANET) security and IPv6 security. He can be contacted at email: alshareeda022@gmail.com.