sustainability

MDPI

Article

Software-Defined Networking Approaches for Link Failure Recovery: A Survey

Jehad Ali [1], Gyu-min Lee [1], Byeong-hee Roh [1,*], Dong Kuk Ryu [2] and Gyudong Park [2]

[1] Department of Computer Engineering, Ajou University, Suwon 16499, Korea; jehadali@ajou.ac.kr (J.A.); mybrand@ajou.ac.kr (G.-m.L.)
[2] Agency for Defense Development, Seoul 05771, Korea; dkryu@add.re.kr (D.K.R.); iobject@add.re.kr (G.P.)
* Correspondence: bhroh@ajou.ac.kr

Received: 7 April 2020; Accepted: 21 May 2020; Published: 22 May 2020

Abstract: Deployment of new optimized routing rules on routers are challenging, owing to the tight coupling of the data and control planes and a lack of global topological information. Due to the distributed nature of the traditional classical internet protocol networks, the routing rules and policies are disseminated in a decentralized manner, which causes looping issues during link failure. Software-defined networking (SDN) provides programmability to the network from a central point. Consequently, the nodes or data plane devices in SDN only forward packets and the complexity of the control plane is handed over to the controller. Therefore, the controller installs the rules and policies from a central location. Due to the central control, link failure identification and restoration becomes pliable because the controller has information about the global network topology. Similarly, new optimized rules for link recovery can be deployed from the central point. Herein, we review several schemes for link failure recovery by leveraging SDN while delineating the cons of traditional networking. We also investigate the open research questions posed due to the SDN architecture. This paper also analyzes the proactive and reactive schemes in SDN using the OpenDayLight controller and Mininet, with the simulation of application scenarios from the tactical and data center networks.

Keywords: software-defined networking; link failure recovery; restoration; resilience; fault-tolerance

1. Introduction

Software-defined networking (SDN) separates the control plane from the data plane, i.e., it moves the control logic from the network devices to a central controller. The centralized controller manages the flow of data through a southbound application programming interface (SB-API). Similarly, a centralized management of the networking devices has the advantage that new applications, services, and networking functions can be flexibly deployed with minimum operating and capital expenses. A few survey studies on SDN operation, history, architecture, programmability, and research directions are described in [1–6].

Link failure recovery approaches leverage the SDN unique features of centralized control and the flexibility of programmable data plane for real-time applications such as video conferencing [7] and voice over IP (VOIP), which can tolerate a delay of 50 ms in case of recovery. Thus, the quality of service (QoS) can be maintained in case of a link failure to ensure untroubled and constant communication. The reported mean for link and device failures in a traditional data center network per day have been recorded as 40.8 and 5.2 failures per time unit [8], respectively, which necessitates the discovery of a method that enables faster recovery of failed links. The study [8] also reported that the frequency of link failures is higher than that of the node failures. Therefore, fault-resilient approaches play an important role in traffic engineering for operator networks to ensure a fast failure recovery, which will ultimately accomplish the requirements of the end-users.

The tight coupling of control and data planes in legacy networks makes them sluggish and complex to manage. Although traditional networks have been adopted universally, their management and configuration are cumbersome [9] because of the following reasons:

- Vendors are hesitant in providing the source code of the protocols to the developer and user community because of being afraid of unverified changes to their devices that can lead to malfunctions in the networks [10].
- A global view of the network is hard to obtain in the traditional network architecture; hence, only distributed routing protocols can be used, e.g., routing information protocol (RIP) [11] and open shortest path first (OSPF) [12].
- The co-existence of data and control planes also leads to an improper utilization of the bandwidth [13], as it is shared by both the planes. Thus, the packets are broadcasted to the network, which leads to low link utilization. Similarly, the ball game gets worse as soon as there is a link failure because the system tries to search alternate paths in the network for packet broadcasting, leading to network congestion.

In case of a link failure, re-routing is performed for discovering an alternative path to divert the packets from a failed link to the alternative path. However, the implementation of traditional routing protocols hinders the network growth and causes delays owing to several problems, such as flooding of the link-state information, long convergence time of path detection [14], deployment complexity of the network [15], and route flaps caused by prefix instability [16]. Additionally, there may be network destabilization because of routing conflicts owing to the autonomous system (AS) [17]. Consequently, there is a lack of optimum decisions due to the unavailability of the global statistics of the network. These problems exist in traditional internet architecture because of two reasons: First, because implementing changes in the traditional routing protocols is difficult owing to the software being embedded in the firmware; and second, the internet companies feel at risk and shy away from implementing any new proposals, even if it can increase the performance of the network, as this will also increase the network complexity and, consequently, the maintenance cost.

Fast failure recovery within a fixed time interval is vital for providing a service guarantee in next-generation technologies. In literature, several architectures [18–20] have been proposed for enabling the fast recovery of networks. The architecture proposed in [18] consists of an automatic failure recovery or fault management framework. The research conducted in [19] leverages 5G, secure Internet-of-Things (IoT), and unmanned aerial vehicle (UAV) swarms to ensure service in mission-critical infrastructures. Likewise, a platform for virtualization of services based on SDN and network function virtualization (NFV) was proposed in [20], which enables the development, implementation, and functioning of media services over 5G networks.

Moreover, the nodes may operate in remote and harsh environments with a possibility of frequent failures. Therefore, consecutive changes are essential to discover an alternative path for the nodes that have experienced failure [21]. In addition, the SDN handles the link failures using one of two main approaches, proactive and reactive [22]. In the proactive approach, the alternate paths are preconfigured, and in the case of a link failure, the disrupted flows are forwarded to the backup path. In contrast, in the reactive scheme, the controller is approached for finding an alternative path and the flow rules for the new path are inserted when the controller calculates the path. The SDN controller, which has access to the global topological information, will search the optimum alternative path for the failed link and will push the flow rules to it. Hence, the data plane is not interrupted. Consequently, the packets, are not broadcasted to the network here due to the centralized control architecture, which leads to a performance improvement in the network. However, both schemes have their pros and cons along with a trade-off in performance and efficiency.

Link failure recovery in SDN was overviewed in [23,24]. In this survey, we investigate the link failure detection and recovery approaches in SDN. A demonstration of the SDN-based failure recovery with proactive and reactive approaches is presented with pictorial diagrams. We compare the proactive

and reactive schemes in terms of latency, scalability, routing updates, ternary contents addressable memory (TCAM) space, flow operations matching, configuration, robustness to backup path failures, routing information access, processing of switches, as well as the routing, controller, and switch overheads. The research issues in SDN-based link failure recovery schemes for large-scale, hybrid, inter-domain, in-band, and machine learning (ML) approaches are discussed and summarized. We simulate two application scenarios in a Mininet testbed for Naval tactical and datacenter networks (DCN) and evaluate the recovery time and throughput when using the proactive and reactive schemes.

The rest of the paper is organized as follows. In Section 2, an overview of SDN architecture is presented and the importance of SDN for achieving the recovery is explicated. In Section 3, we discuss the various link failure detection techniques. In Section 4, the two most common methods for searching the alternative paths, i.e., proactive and reactive, are described, in addition to failure recovery approaches in large-scale networks, inter-domain architecture, hybrid SDN, in-band environment, and ML-based techniques. In Section 5, we discuss SDN application scenarios, experimental setup and an experimental demonstration of the proactive and reactive approaches in Naval tactical and DCN operations. In Section 6, a summary is provided based on the findings in various research studies. Finally, in Section 7, we conclude the paper and highlight the main points of the survey.

2. SDN Architecture

2.1. An Overview of SDN

Figure 1 [25] shows the different layers of the SDN architecture and the way the devices in the data plane interact with those in the control plane through an SB-API. The SB-API provides an interface for interaction between the data and control planes. Several protocols are available for the interaction of the two planes, such as OpenFlow and Netconf [26]. The control plane is implemented through SDN controllers, e.g., POX [27], OpenDaylight (ODL) [28], open network operating systems (ONOS) [29], Floodlight [30], and RYU [31]. The northbound API (NB-API) is an interface between the control and management planes. The SDN controller acts as a bridge between the management and data planes leveraging the representational state transfer (REST) API. Similarly, the statistics about the data plane such as the flows are gathered through REST API.

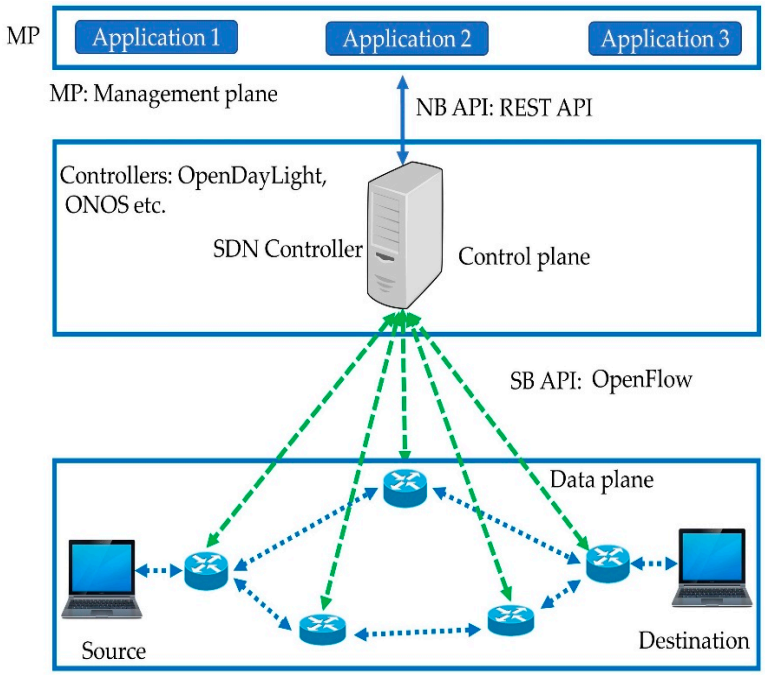

Figure 1. An overview of SDN (software-defined networking) architecture.

The programmability and orchestration in SDN, which is at the top of the control plane, are implemented through the management plane. Different applications are executed for performing the versatile and dynamic operations needed for an efficient link failure recovery. The switches in the data plane leverage the flexibility and programmability features of the management plane through the abstractions offered by the control plane. For example, network monitoring and failure detection can be thoroughly performed by developing and deploying snipping and failure detection applications in the management plane.

2.2. A Global View of the Network

There is a large recovery delay in classical IP networks owing to the flooding of packets, the increased time for failure detection, computation of alternate paths, and updating of the routing table. However, in SDN, the controller has a global view and control of the network. Therefore, it decides optimally while searching an efficient alternate path for the disrupted flows. In addition, the controller monitors the end-to-end connectivity, and therefore, when a link is broken, the controller can reconfigure the network to re-establish the end-to-end (E2E) connectivity for all paths.

In contrast to traditional networks where every node floods the network with packets to find an alternate path, the SDN provides the solutions with less complexity and flexibility. The programmability and flexibility can be used to dynamically apply policies in the network through the control plane, according to the changing QoS requirements as soon as the link failure occurs. Consequently, the cost, time, and workforce is reduced.

2.3. The Low Complexity of Data Plane

SDN shifts the data plane complexity to the centralized controller; therefore, the nodes react efficiently by utilizing the SBI (e.g., OpenFlow/Netconf) unique features for automatic failure recovery and load balancing. Additionally, applications can be executed with more flexibility in the data plane, resulting in improved QoS and quality of experience (QoE), and the fulfillment of the requirements of carrier-grade networks (CGNs). Furthermore, OpenFlow provides fast detection mechanisms that reduce the link failure recovery delay and packet loss. Similarly, backup paths can be easily configured on the switches, and the match and forwarding features can help achieve high network throughput and QoS.

3. Link Failure Detection Mechanisms

The failure recovery process in SDN starts with a detection of the failed links. If the detection is swift, then the overall delay of the recovery will be small, which is why detection schemes are so crucial to the overall process. Table 1 gives an overview of the link failure detection mechanisms, the detection methodology for the failed links, and the related problems in the detection schemes. Modern-day networks leverage the SDN centralized control and flexibility of managing the data plane in the link failure recovery. A global view of the network, facilitated with central control, provides several possibilities in the failure identification process [32–34]. The schemes proposed in [32,33] used the concept of monitoring cycles to reduce the link failure recovery time. The monitoring circle paths detect and locate link failures in the SDN. However, the problem of minimizing the number of hops and monitoring cycles is vital. Therefore, in Reference [32], a binary search technique was introduced to minimize this overhead. Similarly, in Reference [33,34], the monitoring assignment was formulated as the postman problem with a heuristic method [35] for the assignment of the monitoring cycles. However, the detection process was still observed to be slow.

Table 1. An overview of link failure detection schemes in SDN.

Failure Detection Schemes	Mechanism for Detection	Problems
Circle paths monitoring [32,33]	Monitoring cycles	Minimizing the number of hops on the paths
Monitoring cycles [34,35]	Heuristic	Detection process is slow
OpenFlow based [36,37]	Heartbeat messages	Cannot ensure a delay of <50 ms
STP or RSTP [38]	Port status updating	Cannot ensure delay for modern technologies
Bidirectional forwarding detection (BFD) [36,39–41]	Exchange of hello packets	Large number of hello packet exchanges
FDLM [42]	Heartbeat messages	Cannot ensure a delay of <50 ms
MPLS BFD [43,44]	Packet generators	Large number of packet generators
Recovery in smart grids [45]	Packet tagging/BER threshold monitoring using OpenFlow	OpenFlow limitations
SDN-EON [46]	BER threshold monitoring/alarms	Applicable for optical networks only
SFD [47]	Packet loss ratio	Limited to node failures

Quick detection of the failed links improves the performance of the link failure recovery techniques; therefore, the failed links must be detected efficiently before the recovery process. Several mechanisms are available for link failure detection; a few of them are mentioned in [36]. The OpenFlow implementations use the same tool of heartbeat messages for detection as that in Ethernet. A heartbeat message is exchanged between the nodes at regular time intervals, which determines the status of the network. The liveliness is checked by the rate of exchange of hello packets between the nodes; therefore, if a node does not receive a hello packet within the regular time interval of 16 ± 8 ms, the controller is notified about the failed link. If a node does not receive a response within the time of 50–150 ms, the link is considered disconnected [37]. Due to the slow detection rate, the Ethernet cannot meet the CGNs delay demands (< 50 ms).

The spanning tree protocol (STP) [38] and reverse spanning tree protocol (RSTP) have also been used on data link layers for link failure detection. However, their detection period spans in seconds and cannot guarantee the delay requirements of modern technologies. Similarly, OpenFlow fast failover (FF) group [1] and Bidirectional Forwarding Detection (BFD) [39] are also routinely available in the SDN community [36,40,41] for link failure detection and recovery. Additionally, a failure detection mechanism known as failure detection service with low mistake rates (FDLM), which uses heartbeat messages to minimize the errors in detection, was proposed in [42].

The failure detection method for transport networks in the E2E path, described by Kemp et al. [43], used multiprotocol label switching (MPLS) BFD [44]. The scheme utilizes packet generators implemented in the switches by sending probe messages along with the data packets. A link failure is detected when there is a gap between consecutive probe messages. The proposed methodology was able to achieve scalability because different network elements can utilize identical packet generators, which can be separated later through MPLS.

A failure detection approach using a counter mechanism based on the outgoing packets was proposed in [45]. The flow rules installed on the link were tagged and monitored, and the packets were then counted at the destination. The error rate calculated by looking at the difference between the sent and received packets was compared with a threshold value. For a given link, if the error rate exceeded the threshold value, the link was assumed to have failed.

The SDN elastic optical network (SDN-EON) is described in [46], which also uses a threshold mechanism using the bit-error-rate (BER). An OpenFlow agent deployed on the data plane nodes periodically monitors the BER. The BER ratio is compared with a threshold to decide whether the link or node on the path has failed. In case of failure, an alarm message is sent to the controller.

The failure detection scheme at the switch level [47], known as switch failure detection (SFD), uses the failed link and the network topology as an input. To identify a failed switch, the algorithm first finds the source and destination of the failed link. Then, it discovers all the hosts connected with the switch and computes whether the packet loss ratio is 100%. If so, the switch is assumed to have failed. Thus, the failed switch is identified as the one initiating the recovery process.

In the current study, we elaborate on the use of these schemes in SDN. The utilization of each failure detection scheme depends on the particular demands of the network where it will be used. For example, the circle paths monitoring [32,33] and monitoring cycle failure detection [34,35] methods are slow due to the presence of many hops on the E2E path and the heuristic algorithm. Hence, these schemes are not favorable for networks with low latency requirements such as CGNs. However, the OpenFlow [36,37], STP or RSTP [38], and FDLM-based [42] schemes that leverage the periodic heartbeat messages and port status updates are relatively faster. The BFD [36,39–41] and MPLS BFD [43,44] approaches are comparatively fast in failure detection. Hence, modern technologies can leverage these schemes because of their low latency. A few schemes [45,46] are limited to particular networks in SDN, such as link failure recovery in smart grids, SDN-EON, and node failures only.

4. Link Failure Recovery Approaches

The various link failure recovery approaches in SDN can be broadly divided into two categories: proactive and reactive [22]. To understand the working of these two techniques, a background knowledge of the data and control planes, as well as how these planes interact with each other, is necessary. Therefore, we have described the SDN paradigm here in detail. Table 2 lists the classification of various link failure approaches with respect to reactive (R) and proactive (P) mechanisms.

Table 2. Classification of link failure recovery approaches.

Papers Reference	Proactive (P)	Reactive (R)	Large-Scale		Hybrid		Inter-Domain		In-Band		Machine Learning	
			P	R	P	R	P	R	P	R	P	R
[48–62]	√											
[63–69]		√										
[70–73]				√								
[74–84]					√							
[85]							√					
[86,87]								√				
[88–91]										√		
[92,93]									√			
[94–98]												√
[99,100]											√	

4.1. Proactive Approaches in SDN for Link Failure Recovery

As discussed earlier, when a link failure event occurs in the SDN, it is handled either with a reactive or proactive approach. During the normal operation of the network, the flows are forwarded from source to the destination node on the primary path. However, when a link fails, the two schemes handle the failures differently. In proactive recovery, the alternate backup paths are preconfigured. Therefore, the detection normally takes place locally and the flows from the failed link are forwarded to the alternate path immediately without interacting with the controller. Figure 2 shows the management of a link failure with the proactive mechanism. When the link in path#1 fails, the flow rules for the backup path are already configured on the switch. Hence, the packets from the failed link are redirected to the alternate path defined on the switch. The advocates of proactive recovery claim that proactive recovery is more efficient in terms of recovery time because the paths are preconfigured and the control plane need not be consulted for finding an alternative path. Consequently, the stipulation for CGNs can be fulfilled, i.e., the paths are recovered in less than 50 ms. The proactive approach, therefore, provides faster recovery without controller intervention. Consequently, any delay caused by consulting the controller and finding an alternative path will be minimized. However, in the proactive approach, an alternate path for the failed link must be configured for every flow on the failed link, which is not only impractical but may also exceed the flow table entries limitation of the data plane switches. Furthermore, the proactive scheme will also entail additional processing for matching the

incoming flows with numerous flow entries because of the additional flows for the alternate paths. In the following subsections, we investigate several momentous proactive approaches in SDN:

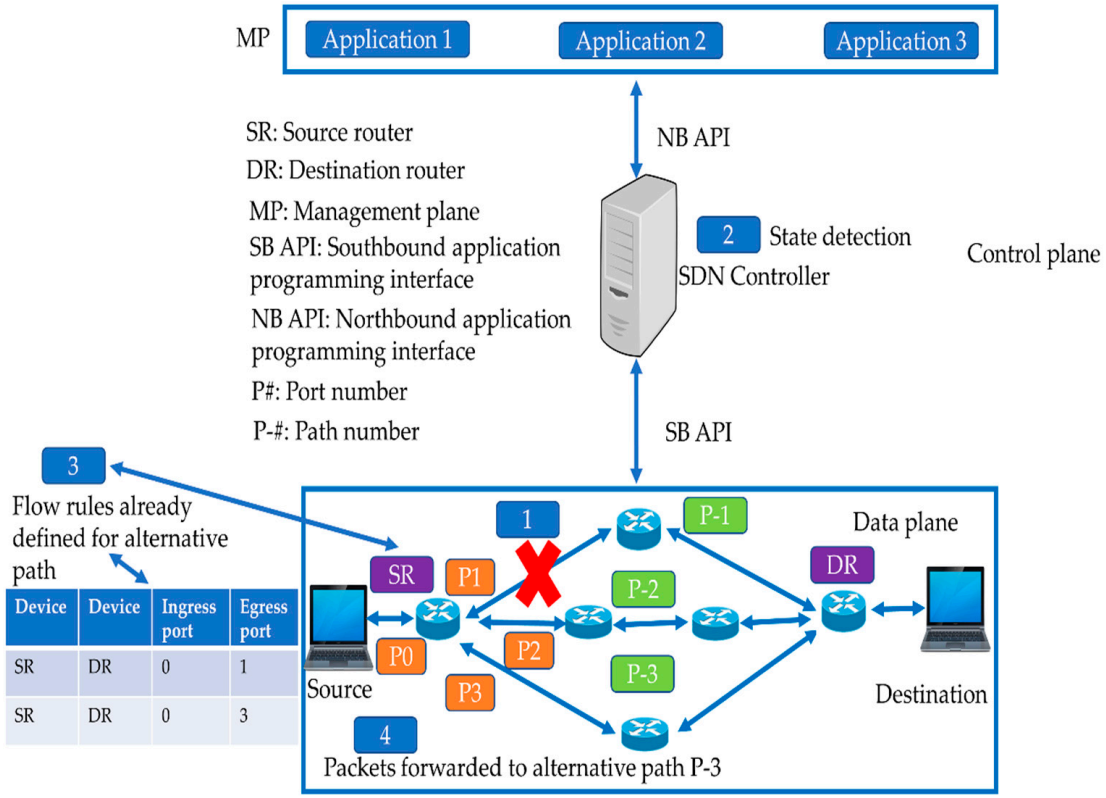

Figure 2. Proactive link failure recovery in SDN.

4.1.1. Link Status Detection with OpenFlow

OpenFlow version v1.1 and onward supports group table functionality, which overcomes the limitations of the flow table entries. Hence, complex packet operations that are not possible through flow entries can be performed. Among the four group types, i.e., ALL, SELECT, INDIRECT, and FAST-FAILOVER (FF), the last type is specifically used for the detection of port failures. Group tables are used to perform a set of actions defined in an OpenFlow bucket. Thus, each group table has a list of buckets as shown in Figure 3. The liveliness of the port/group is monitored from the watch group/port included in the bucket. When a link failure occurs, the status of the bucket watch group/port is flagged as down; therefore, another bucket whose status is up is used. Consequently, only one bucket will be in use at a time. Therefore, the disrupted flows are routed to the alternative path without consulting the control plane, i.e., this is a recovery approach in the data plane.

This approach of utilizing the FF group has been used by several researchers [36,40,41,44]. The FF group can work only if a local secondary or backup path exists at the switch which detects the failure; even if such a path does not exist, the controller must be contacted to find an alternate path to reroute the packets. However, the advocates of data plane recovery solutions contend that recovery should be attempted in the data plane to minimize the recovery delay.

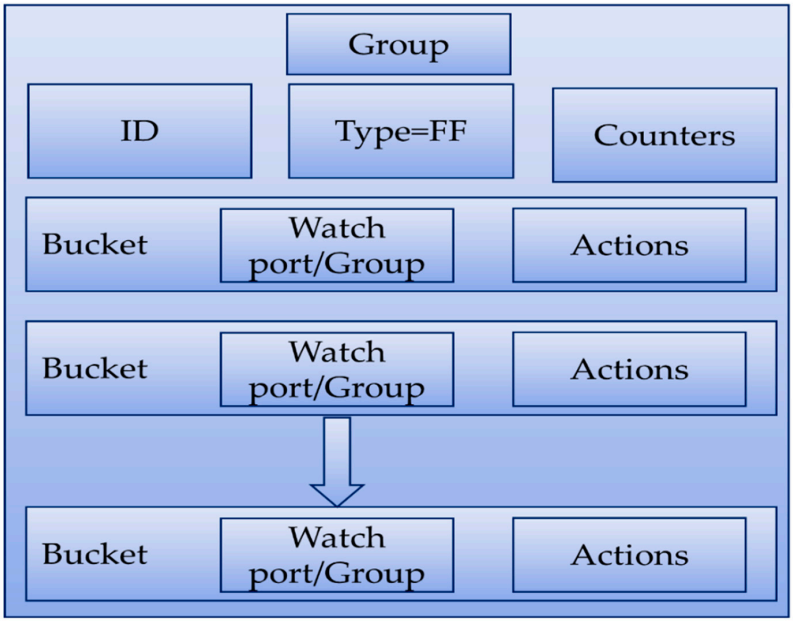

Figure 3. The FF (FAST-FAILOVER) group is designed to detect and respond to port failures.

4.1.2. BFD

The BFD protocol was adopted in SDN from a concept used in legacy technologies that detects the liveliness of the paths, i.e., detects failed links between two endpoints through packet loss detection via an echo and control messages. The link's current state is monitored by checking the control messages transferred by each node. The nodes receiving the control messages send the status of their sessions through echo messages. The schemes that utilize the FF group do not need to consult the controller for link failure recovery after the initial configuration. However, the alternate paths cannot be used in case the first backup path fails if they have not been preconfigured.

4.1.3. SPIDER

A project named SPIDER, which fills the deficiencies in the FF group, was presented in [48]. SPIDER recovers the failed links without consulting the controller in case of alternate path unavailability. SPIDER uses link probing for the detection of failed links and can re-route the packets quickly even without controller availability in case of failures. Hence, it is a complete data plane solution.

4.1.4. Path Priority-Based Scheme

A meek proactive solution described in [49] provides a backup path, where each SDN switch has two flow entries for the path associated with an incoming packet to the switch: an active path and an alternate path in case of a failure. When a failure is detected, the packets are traversed to the backup path associated with the switch. However, the scheme uses a small network emulated with Mininet. Therefore, the solution is not feasible for large-scale SDN due to TCAM space limitation caused by increased match and forwarding actions of the switch.

4.1.5. Congestion-Aware Proactive Schemes

Congestion is an important factor while searching an alternate path for the failed link. Any congestion in the alternate recovered path will cause a loss of packets. Therefore, a congestion-aware scheme was provided in [50], which proposed a data plane proactive recovery technique where a backup path is preconfigured for every active path. All the disrupted flows are re-routed through the backup paths for a failed link through source routing. Although several schemes described in the literature [51–53] deal with combating the congestion problem, where multiple objectives were

proposed, such as (1) recovering the local data plane without involving the controller, (2) reducing the number of flow entries, (3) considering the congestion while re-routing the traffic for the failed link, and (4) recovering the single link as well as the single node. To evade congestion, the source routing explained in [54–56], tags per-hop, and flow spreading were implemented. The proposed scheme effectively achieved a congestion-aware failure recovery with minimum overhead and fulfilled the four objectives.

4.1.6. Challenges in Proactive Recovery

- **TCAM flow entries limitation:** SDN switches have a limitation on the number of entries in their flow tables. State-of-the-art switches in the market can store up to 8000 flow rules. Therefore, the cost for TCAM [57] space can increase.
- **Process of matching:** In the proactive approach, the backup paths are preconfigured. Thus, it increases the number of flow entries in the switches, with a greater number of flow entries especially in large-scale networks. As discussed earlier, when a packet arrives at the data plane switch, it is matched with the flow entries to find the destination of the incoming packet. Consequently, this approach affects the process of matching the incoming packets to the switches.
- **Configuration:** Currently, the networks are highly dynamic, and consequently, the configuration changes and link failures in a single day are also high [8,58–61].
- **Large-scale networks:** The proactive approach is not suitable for large-scale networks because of the enormous increase in the number of flow entries as the network scales upward caused by the presence of a preconfigured backup path for each switch in the data plane.
- **Dynamic network conditions:** There is a possibility that the backup path may fail earlier than the primary path owing to dynamic network updates. Therefore, when a link fails, the path configured proactively will not be available for routing the packets on the alternate path.

4.2. Reactive Approaches for Link Failure Recovery in SDN

Reactive failure recovery mainly relies on the SDN controller, as described in [62]. Figure 4 shows a reactive link failure recovery scenario. The following steps are performed for the link failure recovery in case of failure detection.

1. The controller monitors the status of the network by sending periodic heartbeat messages.
2. The controller detects any case of failure.
3. The controller searches an alternate path for the failed link.
4. The controller deletes the old flow entries and adds new flow entries for the updated path in the SDN switches.

Although reactive techniques find the alternate path dynamically, the controller intervention causes a large recovery delay. This is due to the communication overhead between the switches and the controller, the extra time spent for the alternate path discovery, and the insertion of flow entries for the new path. Hence, the opponents of this approach claim that reactive schemes cannot satisfy the delay bounds of the CGNs, i.e., 50 ms.

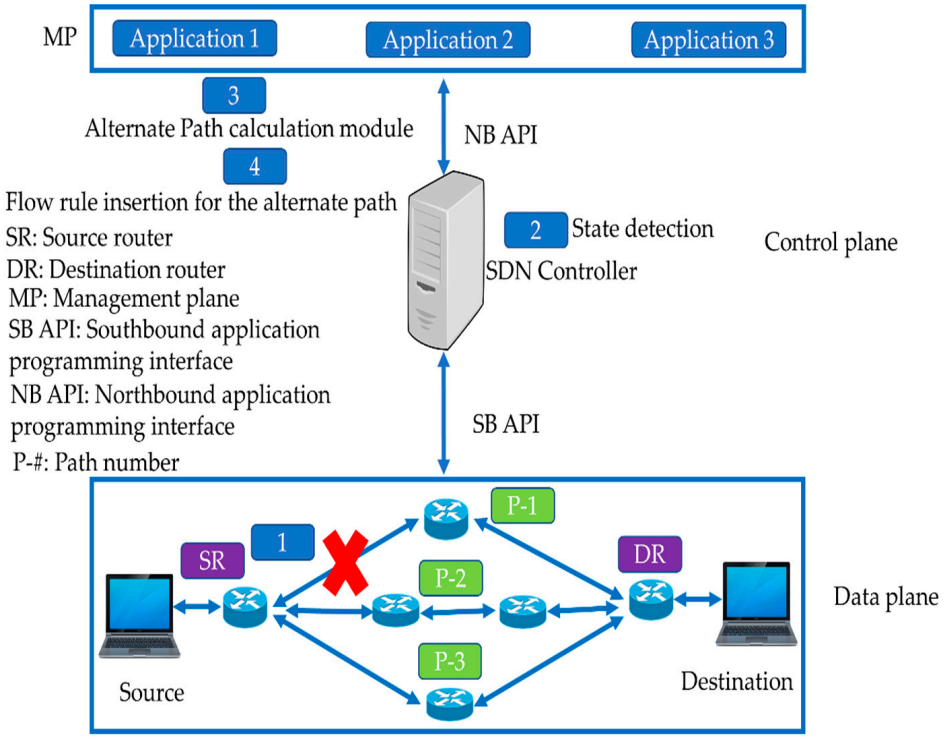

Figure 4. The reactive mechanism for link failure recovery in SDN.

4.2.1. Reactive Approaches with the Shortest Path First

A reactive link failure recovery approach based on the shortest path first algorithm was proposed in [63]. Packets on each path are divided into high and low priority packets. The proposed system ensures minimum delay for high priority packets. However, the approach can only be applied to a small-scale network and is not feasible for large-scale SDNs. The scheme also avoids congestion by distributing the traffic equally over the available paths. Therefore, the complexity of the algorithm increases when the network scales. Another drawback of the proposed method is the insufficient information provided for the implementation mechanism. Additionally, the technique has not been tested on standard internet topology datasets.

4.2.2. Reactive Approaches That Consider the Number of Flow Operations

A delay due to the insertion of flow operations by the controller increases the average recovery delay. Thus, to minimize the flow insertion time, the number of flow operations must be minimized as described in [64]. Consequently, in case of a failure, the selection of an alternative path with low cost will minimize the number of flow operations by the SDN controller, which will consequently lead to an overall reduction in delay. Therefore, the overall link failure recovery delay will be minimum if the number of flow operations from the SDN controller are reduced. A comparison of the issues in proactive and reactive schemes is listed in Table 3.

The SDN controller will also have to install the flow entries for the alternative path. Most of the techniques do not consider the extra delay incurred due to flow operations by the SDN controller in case of link failure recovery. According to this study in Reference [65], the latency induced by the SDN controller due to flow insertion is 0.5 to 10 ms. Similarly, in Reference [66], the authors reported that the minimum time needed for a flow rule addition and deletion is 11 ms. Therefore, in large-scale networks, the minimum delay needed for CGNs (50 ms) cannot be achieved as described in [40]. The authors in [22] proposed that due to this constraint, a delay of 200 to 300 ms is within the realistic range. Hence, the advocates of reactive approaches claim that this delay is not negligible in large-scale SDN failure scenarios and should be minimized because the delay will increase as the SDN scales.

Table 3. A comparative overview of proactive and reactive link failure recovery techniques in SDN.

Issues	Proactive	Reactive
Routing updates	Periodic advertisements	Requested when a failure occurs in the data plane
TCAM space	TCAM space limitation	More TCAM space
Match flow operations	More flow matching	Fewer flow entries matching
Configuration	Hard, because of the large number of backup paths	Easy
Dynamic network updates	No	Yes
Flow entries matching	More, because backup paths exist	Less, because the path is discovered on failure
Switches processing	High, due to more flow matching	Low
Controller overhead	Low	High
Switches overhead	High	Low
Latency	Small, due to preconfigured path	More, due to flow insertion delay
Routing overhead	Proportional to the number of switches in the data plane	Proportional to the number of nodes on the communication path
Routing information access	From the switches in the data plane	With controller interruption, discovery by the routing algorithm
Scalability	Not scalable for large networks	Scalable for large networks
Examples	Provided in [48–61]	Provided in [62–67]

The authors in [67] proposed a single link failure reactive recovery approach based on the graph theory for E2E path computation. A path is split from the source node to half of the total length and then to the destination node. The longest shortest path is then used from the source to the destination. Hence, when a link fails, any other path shorter than the longest shortest path is used for communication. The proposed algorithm in [67] searches the path only between the failed nodes; the remaining flow entries are not removed, which results in minimizing the number of flow operations. Figure 5 shows the longest shortest path between Dublin and Sofia, calculated in the European reference network (ER_Net) [68]. The algorithm returns the set of all paths, i.e., the longest shortest path and the other shortest paths. In case of a link failure between Hamburg and Berlin, the shortest path between Hamburg and Berlin is chosen only between the two failed routers of these cities; therefore, flow entries are inserted only for this portion of the path and the remaining flow entries are preserved. Consequently, the insertion delay of the flow entries is low compared with other schemes that consider the source to destination flow installations for the whole path using the Dijkstra algorithm.

The proposed scheme results in a significant reduction in the number of flow entries by considering an alternate path only between the nodes within the failed link. The algorithm has demonstrated a maximum reduction of 85.69% in the flow entries of the SDN. However, the algorithm neither guarantees the shortest path nor considers the congestion. Thus, the overall E2E delay might be large despite a reduction in the number of flow entries. It is mentioned in Reference [66] that the insertion of a flow entry in a switch takes 0.5 to 10 ms of processing time. Therefore, the overall recovery delay will increase.

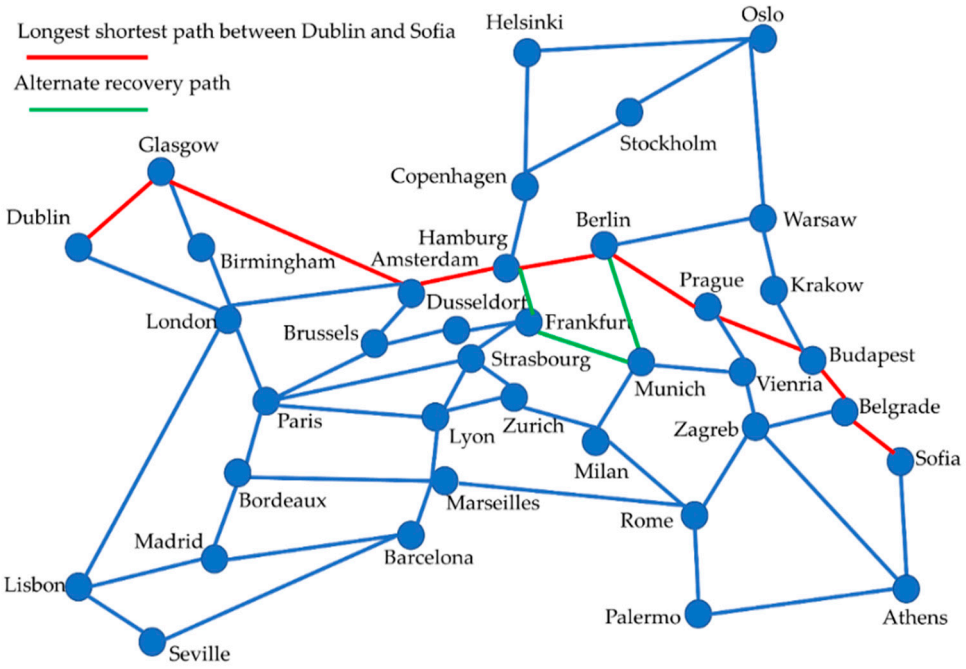

Figure 5. Link failure recovery by minimizing the number of flow operations [67,68].

4.3. Large-Scale SDN

In the case of reactive failure recovery, the SDN controller must find an alternative path after a link failure. Therefore, the existing proposed schemes have focused only on determining the shortest alternative path for the failed flow and neglected the flow insertion time in the flow table of the switch. The proposals presented in [69,70] indicated that restoration approaches that followed the shortest path technique only are not suitable for large-scale SDN and may fail to restore the flows efficiently.

Pruned searching using the graph theory approach was introduced by [71] for a quick recovery of link failures in large-scale SDN. The proposed algorithm shows swift performance even in large-scale networks and guarantees the shortest path. Additionally, the technique has been compared with Dijkstra to demonstrate its lesser complexity. Furthermore, the algorithm implemented in C++ has shown scalability and robustness when tested in a topology comprising over 25,000 nodes and also with some standard internet topologies. Moreover, the idea of prune searching avoids the unnecessary path search operation, which results in lesser complexity of the algorithm and is therefore scalable. However, the author did not discuss the number of flow operations required (addition and removal of flow entries) for link failure recovery.

A distributed controller placement approach has been presented in [72] that utilizes three controllers, one in an active state and two for backup. Therefore, in case of controller failure, the control plane operations continue uninterrupted with less packet loss. According to the findings of the study, the packet loss reduces from 5.22% to 4.15% with distributed controller architecture. Similarly, delay with a distributed control plane reduces from 8.1% to 6.1%.

4.4. Hybrid SDN

In a hybrid SDN, the SDN and traditional routers co-exist simultaneously [73]. According to the authors in [74], a proposal was presented that used a global topology view for controlling the external traffic routes. The routers obtain the route information from the centralized controller. However, the controller cannot gain control of traffic information from legacy devices. A proposal described in [75] utilizes the OpenFlow protocol for obtaining the route statistics from the SDN and legacy nodes. Likewise, the approach in [76] leverages an SDN controller working in parallel with legacy devices for taking routing decisions pretending OSPF link-state advertisements (LSAs). The data and control

plane communication is conducted through various layers of protocol translation. The advocates of hybrid SDN claim that as the SDN is not yet fully deployed, link failure problems related to hybrid SDN must be addressed by taking advantage of the strong features of traditional networks.

A sample architecture for E2E communication in an inter-domain hybrid SDN, illustrated in [77], demonstrated that hybrid SDN leverages border gateway protocol (BGP) functionalities for policy management, path selection, and security. In Reference [78], a rule update consistency mechanism was proposed using legacy and SDN switches for combating security policies. The studies presented in [79–81] proposed link failure solutions for a hybrid environment using legacy and SDN switches. Likewise, the authors in [82] explained various hybrid SDN types, such as those based on class, service, and topology. Similarly, the challenges, opportunities, and trade-off among different types were also discussed in [82].

A congestion-aware approach for link failure recovery in hybrid SDN proposed the redirection of traffic from a failed link to the SDN switches through a pre-established IP tunnel [83]. In the proposed methodology, the switches can also discover multiple backup paths by leveraging the SDN, which helps in avoiding congestion and in balancing the load that was not achieved with the shortest path schemes. Figure 6 shows the general procedure of the approach they followed. However, the proposed method only deals with single link failures.

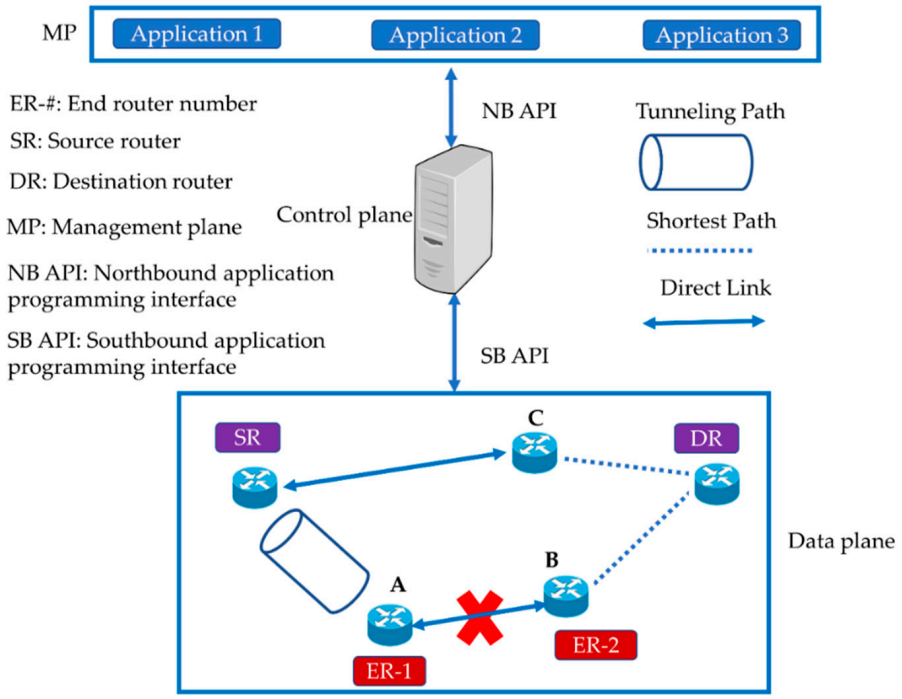

Figure 6. Link failure recovery in a hybrid SDN [77].

4.5. Inter-Domain Approaches

The controllers are an indispensable entity in the SDN, and are utilized either in a central or distributed manner, i.e., either a single domain with one controller or multiple domains with a separate controller for each domain. A single domain controller is only responsible for one AS in its administration. However, the management of different heterogeneous domains is tedious because of a lack of a common northbound interface. Similarly, the network update process may be insecure due to communication between different control planes [84].

Owing to the complexities in the implementation of distributed controller architectures, an SDN approach, i.e., software-defined fast re-routing (SD-FRR) for inter-domain link failures was proposed in [85]. As E2E performance suffers from the slow response shown by BGP [86], it also leads to unreliable delivery of packets. The proposed scheme efficiently recovers the failed link with less

overhead of the controller due to the distributed architecture. Figure 7 shows a hierarchal control plane SDN, with a joint participation of local and global controllers. The local controllers can share the network state information with the global controller. Therefore, the E2E service provisioning can be ensured in case of a link failure. Despite the existence of various proposals with this approach, there are still a few unresolved problems related to interoperability, communication between the control plane and the northbound interface, consistency among various domains, and E2E service provisioning for various applications.

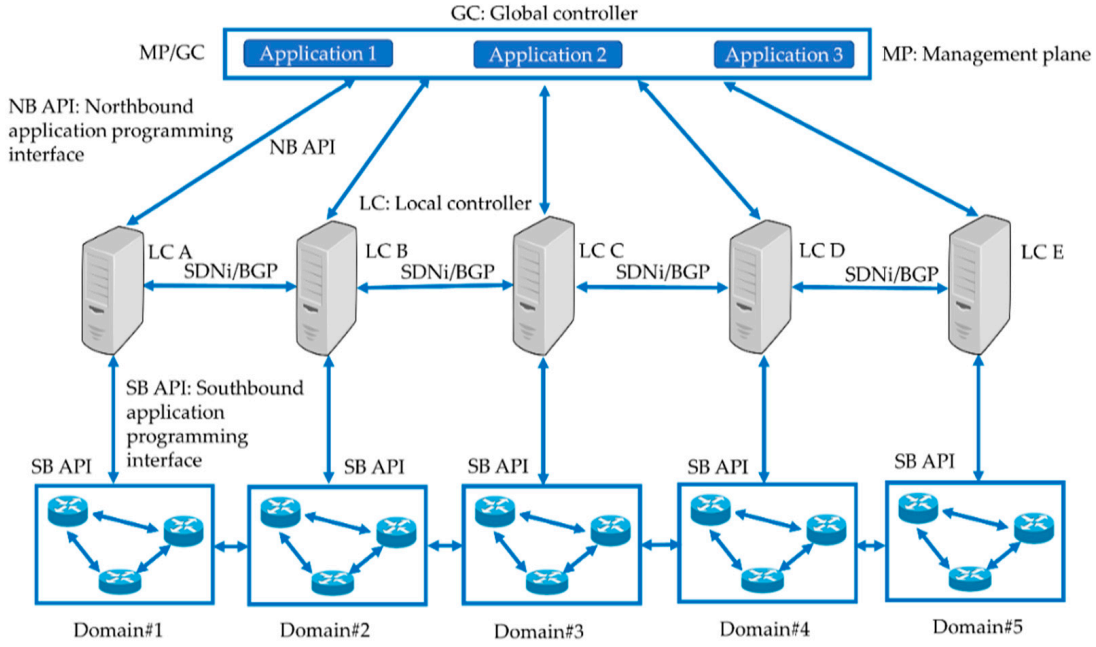

Figure 7. Inter-domain hierarchal architecture for an E2E (end-to-end) communication in SDN.

4.6. In-Band Methods in SDN for Link Failure Recovery

The techniques for link failure recovery discussed so far fit into the out-band SDN methodology, i.e., where the data and control traffic have separate channels for transmission. In in-band SDN, a common channel is used for both data and control traffic. Therefore, the data and control packets compete for shared resources such as bandwidth [87]. In case of a link failure, both the control and data plane traffic are affected. Accordingly, recovery must be initiated for both types of traffic to ensure the smooth operation of the network.

In in-band SDN, in case of an outage in the communication link between the switch and the controller, the switch waits until the echo request message times out [88]. If the connection establishment fails, the switch starts a backup timer and waits for another chance till the timeout of the timer. The same process is repeated for each failure. As the time taken by the echo request and backup timer is 1 s [88], the recovery delay increases. In [89], the authors proposed a scheme for recovery of the control traffic by leveraging the FF group. Using the FF group, the traffic toward a failed link is transferred to an alternative link without interrupting the controller. Action buckets determine the destination of the packet under both normal and failure conditions. During experimental validation, it was observed that the traffic was restored without contacting the controller, which means that it can meet the CGN delay limits (<50 ms). However, the scheme has a single point of failure (SPOF) problem, i.e., a recovery is not possible in case of a controller or switch failure.

An in-band SDN plug and play recovery architecture discussed in [90] is based on the OpenFlow protocol where controllers themselves can create a communication network. Each controller builds a global spanning tree that enables communication among the controllers. Similarly, every switch is configured with controller virtual IP address and a set of rules applicable to the control traffic.

A priority-based scheme described in [91] uses a queuing mechanism with the highest priority given to control traffic in an in-band SDN. Consequently, the data packets do not affect the control packets. However, the synchronization of the data plane switches in SDN with distributed controllers architecture still a problem.

The synchronization of the distributed control plane was tackled in [92], where the authors proposed an in-band synchronization mechanism. Synchronization among the controllers is vital to keep the operations of the data plane consistent between distributed controllers. Multiple configurations in the data plane were combined as an atomic transaction to avoid the conflicts among the transactions. If a transaction leads to inconsistency among the control planes, it is aborted and neither of the control planes executes it. Although the described works provide a consistent and atomic framework for transactions in distributed control planes, the actual implementation of a distributed control plane in SDN still faces several challenges with regard to consistency, reliability, and performance.

4.7. ML in SDN for Link Failure Recovery

The unprecedented growth and complexity in network traffic have placed limitations on the use of conventional recovery schemes in case of link failures. The number of interconnected network devices and the Internet of vehicles data are predicted to cross 50 billion [93] and 300,000 exabytes, respectively, by 2020 [94]. Hence, the probability of link failures will increase in correspondence with the developments in the internet devices. One of the greatest advantages of ML is its ability to provide solutions for complex problems [95]. Similarly, traditional recovery techniques may fail while handling the exchange of large amounts of data. Therefore, the application of ML algorithms to critical traffic attributes can offer useful insights for fault identification and recovery. Consequently, ML algorithms are considered for big data management.

In [96], the authors described the application of ML schemes for self or automated configuration, healing, and optimization. The goal of self-healing is the detection, recovery, and analysis of failures in the network. Herein, it is important to mention that ML schemes are used for the detection and classification of faults. A deep learning-based scheme was proposed in [97] for link failure recovery in 5G networks. Here, the signal strength is measured when the mobile stations move between base stations. The history of past handovers is collected for training the ML model and for prediction of the next handover. Implementing link failure localization in complex networks based on traffic attributes using ML is described in [98]. However, these approaches do not provide an adaptive recovery of the link failures, i.e., no rerouting of traffic is proposed to the destination from a failed to an alternative path after considering traffic conditions.

The availability of data is very critical for ML schemes. In this context, ML algorithms have an advantage that they collect the network traffic data. The SDN controller has a global view of the network, which simplifies the monitoring and collection of network statistics. In Reference [99], an approach was presented for leveraging the SDN features of centralized traffic engineering and network monitoring. The SDN controller monitors the changes in the traffic patterns. Similar to a link failure, there is a change in the traffic pattern on the other paths. The controller updates the secondary path when it detects the change in network traffic. The ML model is trained with various classification algorithms such as support vector machine (SVM), random forests (RF), neural networks (NN), linear regression (LR), and decision trees (DT). However, if the number of nodes increases, then the backup paths will also increase enormously along with an increase in the flow matching overhead. The deep learning-based [97] approaches adapt themselves according to the dynamic changes occurring in the network. Hence, they are ideal for self-organizing networks leveraging the programmability features of SDN controllers.

Table 4 summarizes the issues and disadvantages in an SDN-based link failure recovery schemes for large-scale, hybrid, inter-domain, in-band, and machine learning approaches.

Table 4. Research issues in various schemes for link failure recovery in SDN.

Failure Recovery Scheme	Research Issues	Disadvantages
Large-scale SDN	Flow insertion delay minimization [69,70]	Large recovery delay
	Efficient algorithms for re-routing the packets [71]	Complexity of algorithms running on SDN controller
	Synchronization among distributed controllers [72]	Inconsistency of information
Hybrid SDN	Communication between traditional and SDN-based switches/routers [73–76,78–81]	Interoperability
	Congestion and load balancing [77]	No consideration of a scenario with nil link failures in the network
	E2E synchronization of different domains [83]	BGP limitations on inter-domain co-ordination
Inter-domain	Complex network update process [84]	Distributed control of E2E information
	Increased overhead of SDN controllers [85]	High processing overhead
In-band SDN	Efficient bandwidth utilization [87]	Inefficient utilization of resources
	Increase in recovery delay [88]	High probability of backup timer delay
	Single point of failure [89]	Low performance, network collapse on controller failure
	Synchronization of distributed control plane [90,91]	Separation of control plane traffic from data plane traffic
	Consistency, reliability, performance [92]	Large delay
ML	Limited to traditional networks [95–98]	Vendor interoperability
	SVM, DT, RF, NN, LR [99], the flow match in switches will increase	No assurance of controller service in case of anomalies/attacks

5. Application Scenarios

5.1. Link Failure Recovery in Tactical Networks

Tactical networks are more complex due to their unpredictable environment. A scenario presented in [100,101], also shown in Figure 8, discusses the application of SDN in Naval tactical networks for automatic reconfiguration of the failed links according to the dynamic QoS requirements of the applications. The studies show how the Naval tactical networks can use SDN for maintaining the QoS of applications in a link failure situation. The applicability of SDN makes the Naval tactical networks automatically reconfigurable. An experiment was conducted with the FloodLight controller; the ships exchange the information with each other through a line-of-sight (LOS) link and network operations center (NOC). The controller updates the policies according to the changes in the network. In the next subsection, we define our experimental setup for conducting a link failure recovery based on SDN in Naval tactical and DCN. We record the throughput and failure recovery time for both networks.

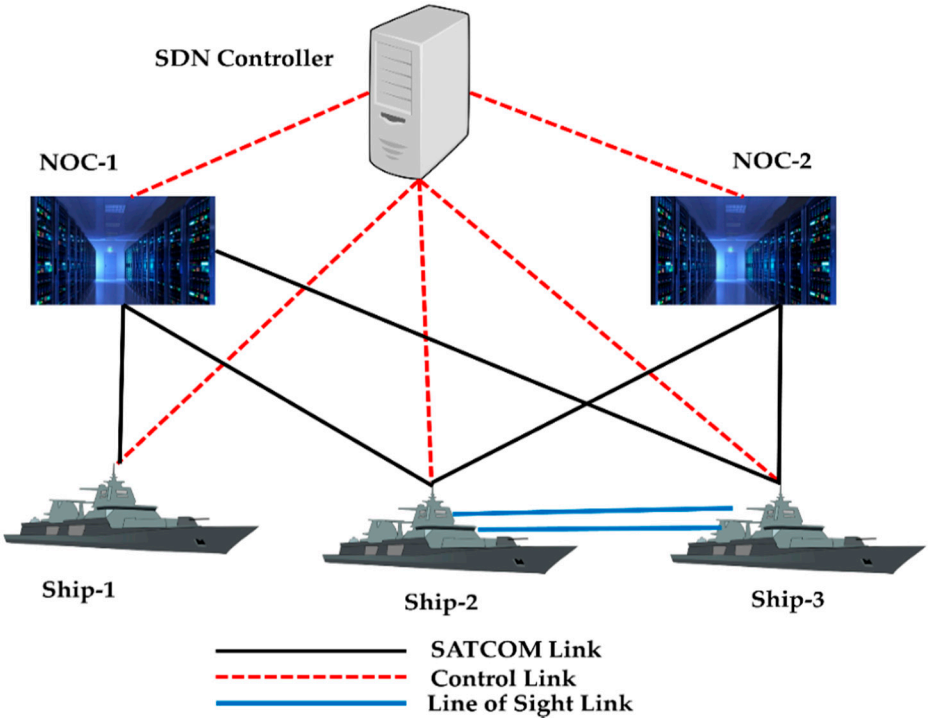

Figure 8. Network topology for Naval tactical networks [101].

5.2. Experimental Setup and Calculation of Recovery Delay

We used the Mininet [102] Python API for emulating the Naval tactical and DCN fat-tree topologies on the ODL [28] controller. The network emulator is widely used for prototyping SDN-based experiments. Mininet version 2.3.0d1 with an Open vSwitch (OVS) version 2.5.4 was installed in Ubuntu 16.04 LTS. Further, the Xming server was used to visualize and generate the traffic between the source and destination hosts. The ODL controller is used in two modes: proactive and reactive. In the proactive mode, the delay or recovery time is equal to the sum of failure detection delay (D_{FD}) and path calculation (D_{PC}), i.e., the time required for finding an alternative path for a failed link. Hence, the flow insertion delay $D_{FI} \rightarrow 0$, as the paths are configured in advance. In the reactive mode, the total delay is the sum of the failure detection delay (D_{FD}), flow insertion (FI) and its delay (D_{FI}), and the path calculation (D_{PC}) from the source to destination hosts. Thus,

$$DP_{total} = D_{FD} + D_{PC} + D_{FI} \rightarrow 0 \quad (1)$$

$$DR_{total} = D_{FD} + D_{PC} + D_{FI} \quad (2)$$

To simulate the link failure scenario, we performed the following steps:

1. First, we configured the ODL controller in the proactive mode and then in the reactive mode.
2. We implemented the Naval tactical and DCN topologies in the Mininet emulator by calling the IP address of the ODL controller.
3. The link bandwidths for the Naval tactical network operation centers (NOC-1 and NOC-2) was 4 Mbps, whereas it was 6 Mbps for the ships Ship-1, Ship-2, and Ship-3.
4. The link bandwidth was 10 Gbps DCN core links and was 1 Gbps for the aggregate links.
5. We then opened an x-terminal (a graphical terminal) on the source and destination hosts.
6. We defined the traffic parameters to be used to evaluate the recovery delay as shown in Table 5.
7. We employed a distributed internet traffic generator (D-ITG) [103] to initiate traffic between the source and destination hosts using the graphical terminals.

8. We induced a link disruption (with link-down command) in the Mininet and recorded the delay using Equations (1) and (2) for the proactive and reactive schemes, respectively.

Table 5. Traffic generation parameters.

Parameter	Payload Length (bytes)	Traffic Rate (Packets/sec)	Time (in Seconds)	TCP/UDP
P1	6000	60,000	100	UDP
P2	7000	70,000	100	TCP
P3	8000	80,000	100	TCP
P4	9000	90,000	100	TCP
P5	10,000	100,000	100	TCP

5.3. Evaluation of Proactive and Reactive Recovery in Tactical (Naval) Network

In this section, we perform a simulation of the topology given in Figure 8 in Mininet. First, we introduced a link failure and then applied the SDN proactive and reactive link failure schemes using the OpenDaylight controller. The experimental scenario and the results validate how SDN automatically reconfigures the network through its programmable interface for forwarding traffic, in case of a link failure depending on the availability of alternate paths. The flow rules updated on the switches determine the forwarding of the traffic on the alternative path to the destination. There are five wide area network (WAN) switches (two NOCs at the shore and three ships). A satellite link connects the NOCs at the shore and the ships. The ships can also communicate via an LOS link depending on the network and distance conditions.

The SDN controller discovers the topology by sending link layer discovery protocol (LLDP) [104] messages to the switches. We induced a link outage in the network and recorded the throughput between the source and destination hosts using the iperf tool [105]. When the link failure occurs due to environmental conditions, the controller is notified by the exchange of LLDP messages between itself and the switch. Ship-2 can communicate via NOC-2 with Ship-3. However, the satellite communications (SATCOM) link between Ship-2 and NOC-2 fails. The SDN controller then re-establishes the path automatically, i.e., it finds an alternative path to re-route the packets between the source and destination nodes. Consequently, a throughput degradation is observed, as shown in Figures 9 and 10. We can observe that there is a larger throughput degradation in Figure 10 owing to controller intervention for the recovery of the link in the reactive approach.

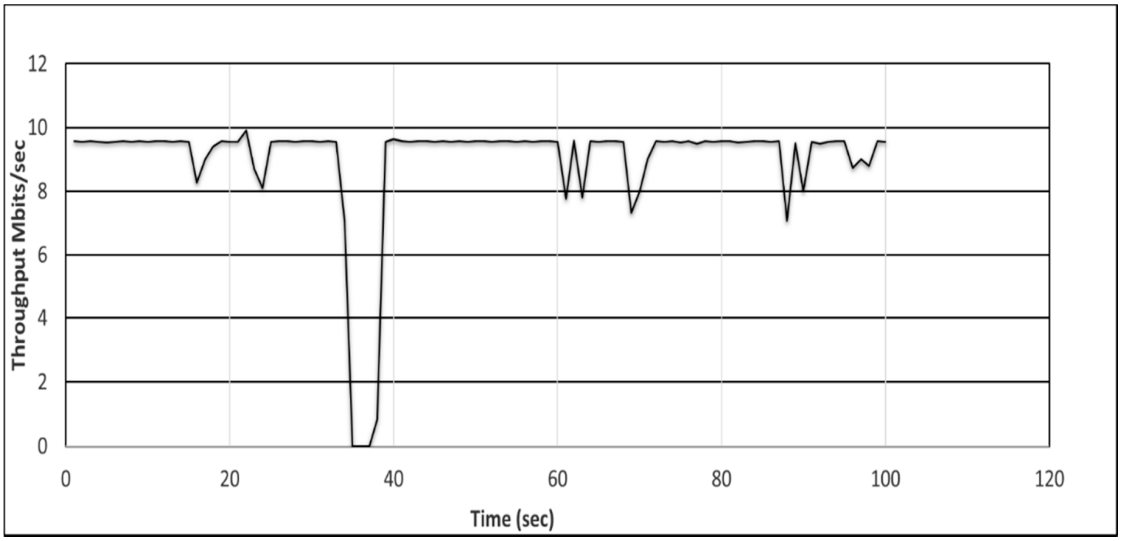

Figure 9. Link failure recovery in SDN with the proactive approach.

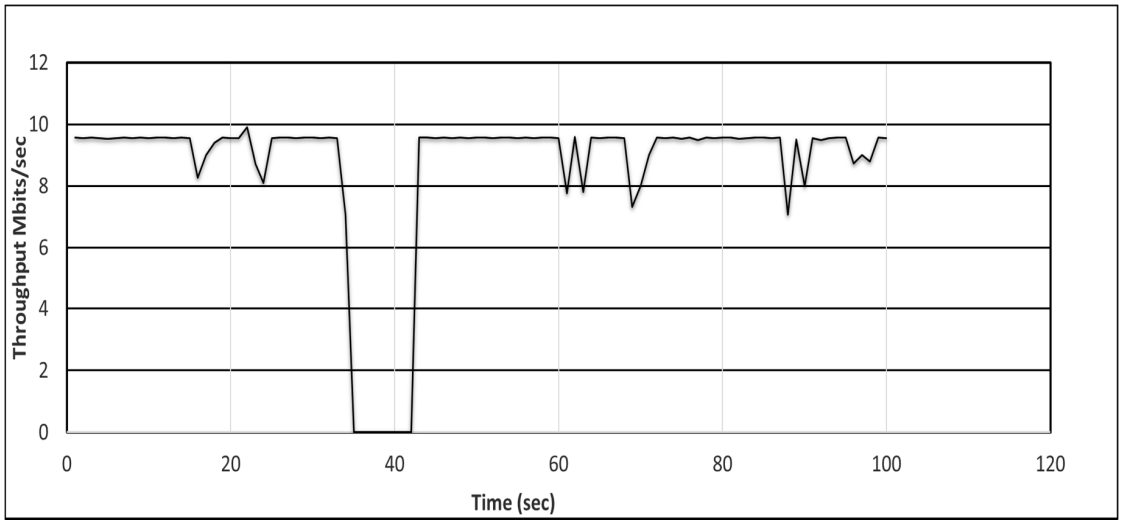

Figure 10. Link failure recovery in SDN with the reactive approach.

Figure 11 shows a comparison of recovery times (in milliseconds (ms)) for the proactive and reactive schemes. We calculated the recovery time for different traffic parameters (P1, P2, P3, P4, and P5) for the topology shown in Figure 8. For this purpose, we used a distributed internet traffic generator tool. The details of these parameters are given in Table 5. We increased the traffic generation rate between the source and destination hosts and recorded the recovery times for both the approaches. We can see that the recovery time in the reactive approach increases with an increase in the traffic generation rate as additional packets are sent to the controller. Moreover, in case of a link failure, the SDN controller discovers an alternate path and inserts the flow entries in the path. Hence, the recovery time is quicker when compared with the proactive scheme.

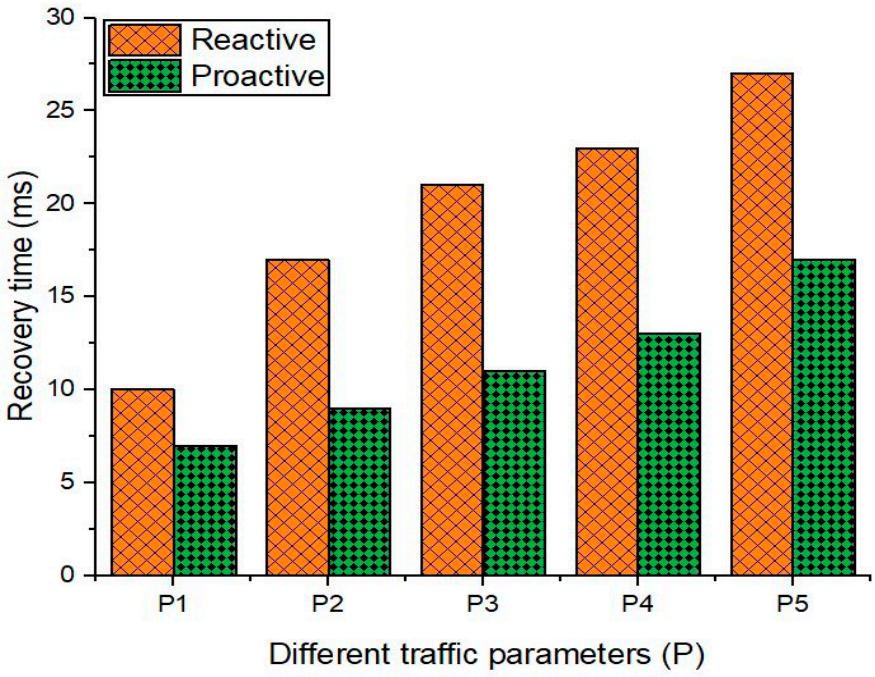

Figure 11. Comparison of recovery times in the tactical (Naval) network with different traffic parameters.

5.4. Link Failure Recovery in a Data Center Network

The mean time for link failures in a data center [8] is 40.8 per day. Therefore, faster recovery of failed links is required to avoid congestion and balance the load. Hence, fast fault recovery approaches

are vital in DCN for fast failure recovery to improve the QoS and QoE. Figure 12 shows a DCN topology with backup paths.

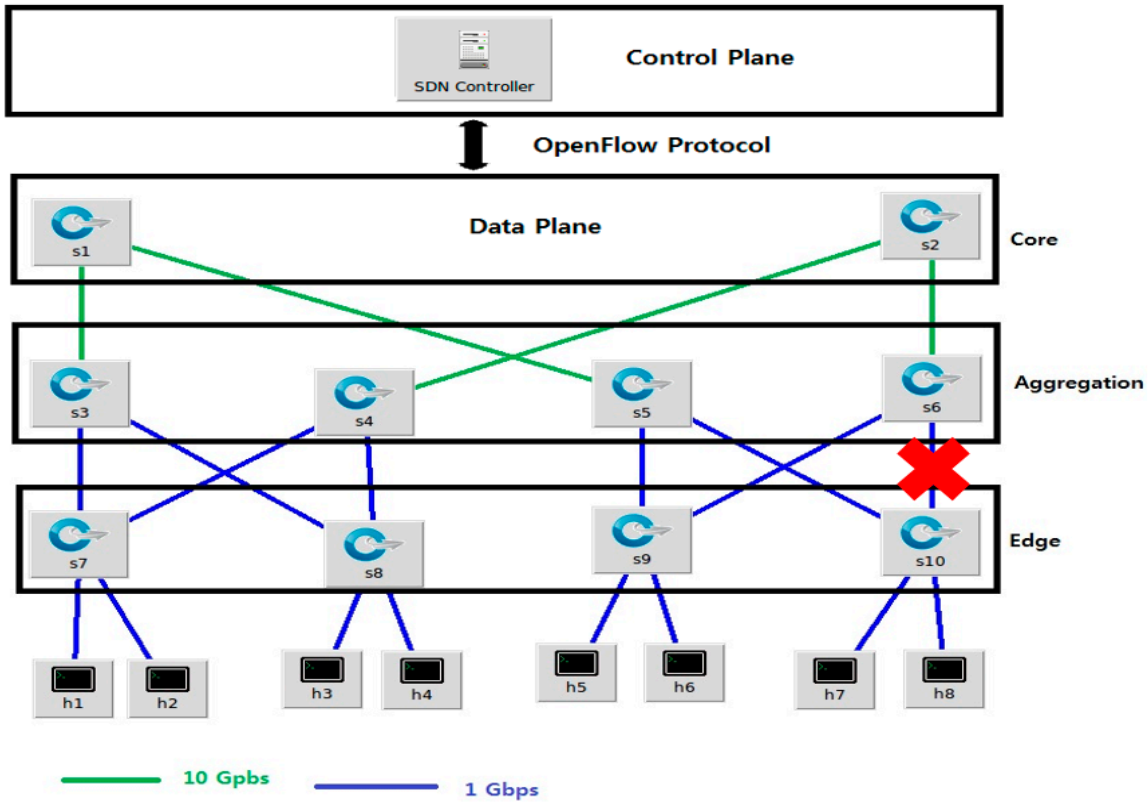

Figure 12. The topology of DCN (data center network) in Mininet.

5.5. Testing Proactive and Reactive Recovery in DCN Using ODL Controller

In this section, we describe the simulation of a scenario for a fat-tree DCN [106], as shown in Figure 12. The ODL controller is configured in two modes: proactive and reactive. The source (h1) communicates with the destination (h8). The link bandwidth is shown in the figure with green (10 Gbps) and blue (1 Gbps) lines. Then, we induced a link failure, as shown with a red cross sign. The path with the red cross sign shows the original link through which the packets are passing toward the destination. The throughput before and after the recovery with the two schemes, proactive and reactive, is shown in Figures 13 and 14, respectively. In both the proactive and reactive approaches, the throughput is almost the same after recovery. However, the average throughput in the reactive approach is smaller and the time for recovery is larger when compared with the proactive approach, because of controller intervention, i.e., an alarm is sent to the controller immediately after a failure. The controller then runs the algorithm for discovering an alternate path for the failed link. Therefore, the observed throughput is comparatively smaller in the reactive approach.

Furthermore, we evaluated the recovery times (ms) when using the proactive and reactive schemes in the DCN topology. Figure 15 shows the recovery times (ms) of both the approaches with the various traffic generation parameters listed in Table 5. It is evident from the figure that the proactive scheme has a lesser recovery time due to predefined alternate paths. Hence, it avoids the extra latency caused by the controller owing to the insertion of flow rules and finding of the recovery path.

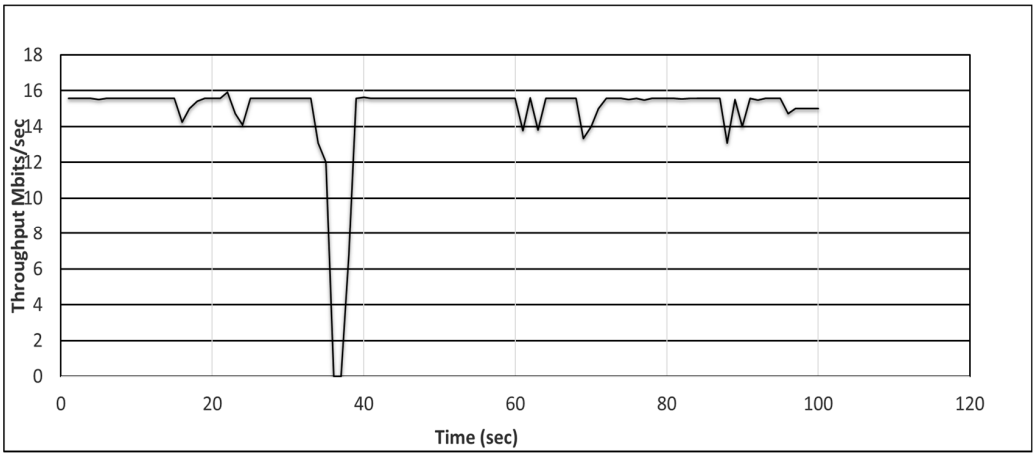

Figure 13. Link failure recovery in DCN with a proactive approach.

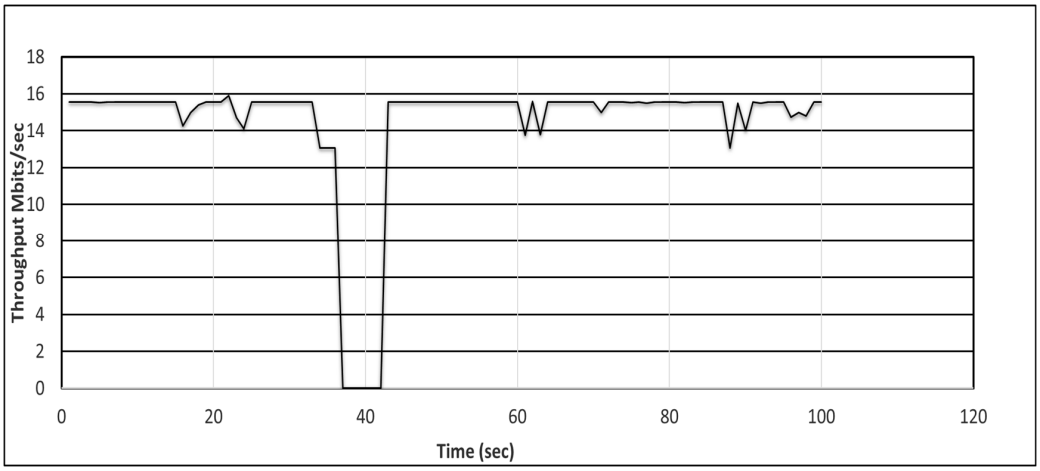

Figure 14. Link failure recovery in DCN with a reactive approach.

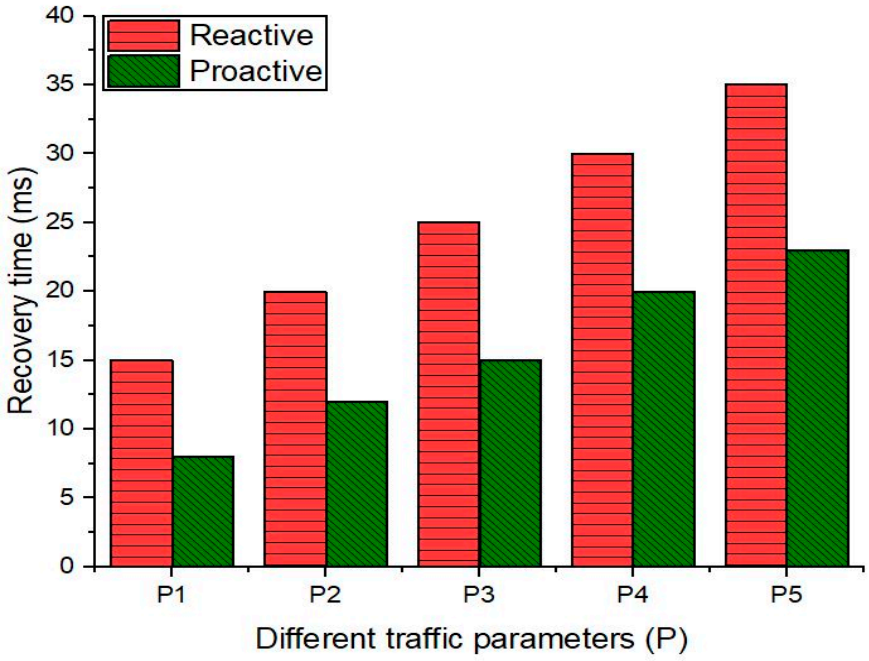

Figure 15. Comparison of recovery times in DCN with different traffic parameters.

6. Summary and Challenges of the SDN-Based Failure Recovery Approaches

In the proactive method, the backup paths are calculated in advance. Therefore, when a link fails, the controller forwards the traffic on the backup path. The method has its pros, such as the controller does not need to recalculate the path as the forwarding rules for the backup path already exist in SDN switches. However, a disadvantage of this approach is that the TCAM space cost of the SDN switches increases. Besides this, the switches have a limitation of 8000 flow entries in the flow tables.

In a few cases, the backup path may fail earlier than the original primary path. If the failure occurs early, the performance is affected, because the incoming packets are matched with the flow rules due to the redundancy of backup path flow entries in the switches. In the reactive approach, the SDN controller installs the flow rules for the alternative path when a link failure event occurs. The methodology is economical in terms of TCAM space; however, the calculation of an alternative path at run time and the installation of rules for the alternative path incurs an additional delay.

To summarize, the critiques of the reactive approach argue that the induced delay incurred by the controller in finding an alternative path cannot meet the minimum delay requirements of the CGNs. However, approaches that have used efficient routing algorithms and minimum flow operations have achieved the desired results. There is always a space for future researchers in terms of improving the previous works because there is a tradeoff between flow operations, large-scale SDN, the minimum shortest cost path, complexity of the algorithm, delay, congestion, load balancing, etc. The inter-domain techniques have synchronization, E2E service provisioning, and interoperability problems that hamper failure recovery. Similarly, in the in-band schemes, the differentiation between data and control traffic is a complex process. Therefore, efficient solutions with minimum complexity can be proposed with which the innovative features of southbound interface protocols, such as OpenFlow/Netconf, can be combined for achieving efficient results. In the end, we discussed ML-based schemes. There is a high probability of the ML-based schemes being used in the future because of the increase in the internet nodes and users as well as the enormous usage of data. However, the lack of standard datasets for the SDN environment hinders the use of ML in SDN research. The development of ML applications with high accuracy for link failure detection and the formation of versatile datasets should be considered for using ML in SDN in future.

7. Conclusions

The introduction of SDN for combating link failure recovery is a novel approach that leverages centralized control concepts. In this paper, we described the background and importance of SDN in link failure recovery by explaining the vulnerabilities of the traditional networking architecture. Then, the three SDN planes and their interaction mechanisms were described along with the importance of SDN for link failure recovery. The failure recovery speed is dependent on the time taken in failure detection. Therefore, we described the state-of-the-art approaches for link failure detection with their pros and cons. We described the proactive and reactive approaches. First, we explained the link failure detection and recovery process with proactive failure recovery in SDN. Then, previous schemes using proactive recovery were described in detail. Similarly, we described reactive failure recovery approaches, i.e., the reactive failure recovery mechanism in SDN and its related literature. We compared the effectiveness of proactive and reactive failure recovery approaches in SDN from the summaries of previous works. A comparison was performed between the proactive and reactive schemes in terms of latency, scalability, routing updates, TCAM space, flow operations matching, configuration, robustness to backup path failures, routing information access, processing of switches, and the overheads of routing, controller and switches. The inter-domain and intra-domain architectures for link failure recovery were discussed. Finally, the link failure recovery in a hybrid SDN environment, large-scale networks, in-band SDN, and machine learning schemes were discussed. We simulated two application scenarios of the Naval tactical networks and DCN using the ODL controller for proactive and reactive approaches, showing the recovery time and throughput comparison. The experimental results after applying the two schemes show that flow insertion by the SDN controller, in case of the

reactive approach, causes an extra burden due to controller intervention. Finally, in the summary section, we listed the problems and future directions for SDN-based link failure recovery. Keeping in view the research challenges, this study can help in selecting the optimum resilient SDN solutions that can guarantee a smooth functioning of the CGNs.

Author Contributions: Conceptualization, J.A., G.-m.L. and B.-h.R.; Funding acquisition, B.-h.R.; Investigation, J.A., G.-m.L., B.-h.R., D.K.R. and G.P.; Methodology, J.A., B.-h.R., D.K.R. and G.P.; Supervision, B.-h.R.; Validation, J.A.; Writing—Original draft, J.A., G.-m.L., and B.-h.R.; Writing—Review and editing, J.A., G.-m.L., B.-h.R., D.K.R. and G.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by C2 integrating and interfacing technologies laboratory of Agency for Defense Development (UD180013ED).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

SDN	Software-defined networking
TCAM	Ternary content addressable memory
API	Application programming interface
NB/SB API	Northbound/Southbound API
E2E	end-to-end
BFD	Bidirectional forwarding protocol
ODL	OpenDaylight
BER	Bit-error-rate
SD-EON	Software-defined elastic optical network
SD-FRR	Software-defined fast re-routing
BGP	Border gateway protocol
CGN	Carrier grade network
FRR	Fast re-routing
LLDP	Link layer discovery protocol
RIP	Routing information protocol
OSPF	Open shortest path first protocol
AS	Autonomous system
STP	Spanning tree protocol
NOC	Network operations center
DCN	Data center network
ONOS	Open network operating system
QoS	Quality of service
QoE	Quality of experience
MPLS	Multi-protocol label switching
RSTP	Rapid spanning tree protocol
FDLM	Failure detection service with low mistake rates

References

1. Lara, A.; Kolasani, A.; Ramamurthy, B. Network Innovation Using OpenFlow: A Survey. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 493–512. [CrossRef]
2. Hu, F.; Hao, Q.; Bao, K. A Survey on Software-Defined Network and OpenFlow: From Concept to Implementation. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 2181–2206. [CrossRef]
3. Ndiaye, M.; Hancke, G.P.; Abu-Mahfouz, A.M. Software Defined Networking for Improved Wireless Sensor Network Management: A Survey. *Sensors* **2017**, *17*, 1031. [CrossRef] [PubMed]
4. Hakiri, A.; Berthou, P. Leveraging SDN for the 5G networks: Trends prospects and challenges. *arXiv* **2015**, arXiv:1506.02876.
5. Singh, S.; Jha, R.K. A Survey on Software Defined Networking: Architecture for Next Generation Network. *J. Netw. Syst. Manag.* **2017**, *25*, 321–374. [CrossRef]

6. Rojas, E.; Doriguzzi-Corin, R.; Tamurejo, S.; Beato, A.; Schwabe, A.; Phemius, K.; Guerrero, C. Are We Ready to Drive Software-Defined Networks? A Comprehensive Survey on Management Tools and Techniques. *ACM Comput. Surv.* **2018**, *51*, 1–35. [CrossRef]
7. Retvari, G.; Csikor, L.; Tapolcai, J.; Enyedi, G.; Csaszar, A. Optimizing IGP link costs for improving IP-level resilience. In Proceedings of the 8th International Workshop on the Design of Reliable Communication Networks (DRCN), Krakow, Poland, 10–12 October 2011; pp. 62–69.
8. Gill, P.; Jain, N.; Nagappan, J. Understanding Network Failures in Data Centers: Measurement, Analysis, and Implications. In *ACM SIGCOMM Computer Communication Review*; ACM: Toronto, ON, Canada, 2011; Volume 41, pp. 350–361.
9. Benson, T.; Akella, A.; Maltz, D. Unraveling the complexity of network management. In Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, NSDI'09, USENIX Association, Berkeley, CA, USA, 22–24 April 2009; pp. 335–348.
10. McKeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.; Turner, J. OpenFlow: Enabling Innovation in Campus Networks. *ACM SIGCOMM Comput. Commun. Rev.* **2008**, *38*, 69–74. [CrossRef]
11. Hedrick, C. *Routing Information Protocol*; Tech. Rep. RFC1058; Rutgers University: New Brunswick, NJ, USA, 1988.
12. Moy, J.T. *OSPF Version 2*, Tech. Rep. RFC2328; IETF: USA, 1988.
13. Sezer, S.; Scott-Hayward, S.; Chouhan, P.; Fraser, B.; Lake, D.; Finnegan, J.; Viljoen, N.; Miller, M.; Rao, N. Are we ready for SDN? Implementation challenges for software-defined networks. *Commun. Mag. IEEE* **2013**, *51*, 36–43. [CrossRef]
14. Oliveira, R.; Zhang, B.; Pei, D.; Zhang, L.; Izhak-Ratzin, R. Quantifying path exploration on the Internet. *IEEE/ACM Trans. Netw.* **2009**, *17*, 445–458. [CrossRef]
15. Caesar, M.; Rexford, J. BGP routing policies in ISP networks. *IEEE Netw.* **2005**, *19*, 5–11. [CrossRef]
16. Mcpherson, D.; Gill, V.; Walton, D.; Retana, A. *Border Gateway Protocol (BGP) Persistent Route Oscillation Condition*, Tech. Rep. RFC 3345; RFC Editor: USA, 2002.
17. Griffin, T.G.; Wilfong, G. An analysis of BGP convergence properties. *ACM SIGCOMM Comput. Commun. Rev.* **1999**, *29*, 277–288. [CrossRef]
18. Vilchez, J.M.S.; Yahia, I.G.B.; Crespi, N.; Rasheed, T.; Siracusa, D. Softwarized 5G networks resiliency with self-healing. In Proceedings of the 1st International Conference on 5G for Ubiquitous Connectivity (5GU), Akaslompolo, Finland, 26–28 November 2014; pp. 229–233.
19. Zahariadis, T.; Voulkidis, A.; Karkazis, P.; Trakadas, P. Preventive maintenance of critical infrastructures using 5G networks drones. In Proceedings of the 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Lecce, Italy, 29 August–1 September 2017; pp. 1–4.
20. Rizou, S.; Athanasoulis, P.; Andriani, P.; Iadanza, F.; Carrozzo, G.; Breitgand, D.; Weit, A.; Griffin, D.; Jimenez, D.; Acar, U.; et al. A service platform architecture enabling programmable edge-to-cloud virtualization for the 5G media industry. In Proceedings of the 2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), Valencia, Spain, 6–8 Jun 2018; pp. 1–6.
21. Akyildiz, I.F.; Su, Y.; Sankarasubramaniam, Y.; Cayirci, E. Wireless sensor networks: A survey. *Comput. Netw.* **2002**, *38*, 393–422. [CrossRef]
22. Akyildiz, I.F.; Su, W.; Sankarasubramaniam, Y.; Cayirci, E. A roadmap for traffic engineering in SDN—OpenFlow networks. *Comput. Netw.* **2014**, *71*, 1–30. [CrossRef]
23. da Rocha, F.; Paulo, C.; Edjard, S.M. A survey on fault management in software-defined networks. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2284–2321.
24. Muthumanikandan, V.; Valliyammai, C. A survey on link failures in software defined networks. In Proceedings of the 2015 Seventh International Conference on Advanced Computing (ICoAC), Chennai, India, 15–17 December 2015.
25. *Software-Defined Networking: The New Norm for Networks*; ONF White Paper; Open Networking Foundation: Palo Alto, CA, USA, 2012.
26. Crawshaw, J. *NETCONF/YANG: What's Holding Back Adoption & How to Accelarate It*; Heavy Reading Reports; New York, NY, USA, 2017. Available online: https://www.oneaccess-net.com/images/public/wp_heavy_reading.pdf (accessed on 7 April 2020).
27. POX Controller. Available online: https://github.com/noxrepo/pox (accessed on 10 January 2020).

28. OpenDaylight. Available online: https://www.opendaylight.org/ (accessed on 3 February 2020).
29. ONOS. Available online: http://onosproject.org/ (accessed on 20 February 2020).
30. Floodlight. Available online: http://www.projectfloodlight.org/ (accessed on 11 January 2020).
31. RYU. Available online: https://osrg.github.io/ryu/ (accessed on 7 March 2020).
32. Kozat, U.C.; Liang, G.; Kokten, K. On diagnosis of forwarding plane via static forwarding rules in software defined networks. In Proceedings of the IEEE Conference on Computer Communications (INFOCOM), Toronto, ON, Canada, 27 April–2 May 2014; pp. 1716–1724.
33. Lee, S.S.W.; Li, K.Y.; Chan, K.Y.; Lai, G.H.; Chung, Y.C. Path layout planning and software based fast failure detection in survivable OpenFlow networks. In Proceedings of the 2014 10th International Conference on the Design of Reliable Communication Networks (DRCN), Ghent, Belgium, 1–3 April 2014; pp. 1–8.
34. Lee, S.S.W.; Li, K.Y.; Chan, K.Y.; Lai, G.H.; Chung, Y.C. Software-based fast failure recovery for resilient OpenFlow networks. In Proceedings of the 2015 7th International Workshop on Reliable Networks Design and Modeling (RNDM), Munich, Germany, 5–7 October 2015; pp. 194–200.
35. Ahr, D.; Reinelt, G. A tabu search algorithm for the min–max k-Chinese postman problem. *Comput. Oper. Res.* **2006**, *33*, 3403–3422. [CrossRef]
36. Van Andrichem, N.L.M.; Van Asten, B.J.; Kuipers, F.A. Fast Recovery in Software-Defined Networks. In Proceedings of the 2014 Third European Workshop on Software Defined Networks, London, UK, 1–3 September 2014.
37. Wang, L.; Chang, R.F.; Lin, E.; Yik, J. Apparatus for Link Failure Detection on High Availability Ethernet Backplane. US Patent 7,260,066, 21 August 2007.
38. Levi, D.; Harrington, D. *Definitions of Managed Objects for Bridges with Rapid Spanning Tree Protocol*; RFC 4318 (Proposed Standard); Internet Engineering Task Force: Fremont, CA, USA, December 2005.
39. Katz, D.; Ward, D. *Bidirectional Forwarding Detection (BFD)*; RFC 5880 (Proposed Standard); Internet Engineering Task Force: Fremont, CA, USA, 2010.
40. Sharma, S.; Staessens, D.; Colle, D.; Pickavet, M.; Demeester, P. Openflow: Meeting carrier-grade recovery requirements. *Comput. Commun.* **2013**, *36*, 656–665. [CrossRef]
41. Lin, Y.D.; Teng, H.Y.; Hsu, C.R.; Liao, C.C.; Lai, Y.C. Fast failover and switchover for link failures and congestion in software defined networks. In Proceedings of the 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia, 22–27 May 2016; pp. 1–6.
42. Yang, T.W.; Wang, K. Failure Detection Service with Low Mistake Rates for SDN Controllers. In Proceedings of the 2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS), Kanazawa, Japan, 5–7 October 2016.
43. Kempf, J.; Bellagamba, E.; Kern, A.; Jocha, D.; Takacs, A. Scalable fault management for OpenFlow. In Proceedings of the 2012 IEEE International Conference on Communications (ICC), Ottawa, ON, Canada, 10–15 June 2012; pp. 6606–6610.
44. Aggarwal, R.; Kompella, K.; Nadeau, T.; Swallow, G. *Bidirectional Forwarding Detection (BFD) for MPLS Label Switched Paths (LSPs)*; RFC 5884, RFC 7726; Internet Engineering Task Force: Fremont, CA, USA, 2010.
45. Gyllstrom, D.; Braga, N.; Kurose, J. Recovery from link failures in a smart grid communication network using OpenFlow. In Proceedings of the 2014 IEEE International Conference on Smart Grid Communications (SmartGridComm), Venice, Italy, 3–6 November 2014; pp. 254–259.
46. Liu, L.; Choi, H.Y.; Tsuritani, T.; Morita, I.; Casellas, R.; Martinez, R.; Munoz, R. First proof-of-concept demonstration of OpenFlowcontrolled elastic optical networks employing flexible transmitter/receiver. In Proceedings of the International Conference on Photonics in Switching, Corsica, France, 11–14 September 2012; p. 5.
47. Muthumanikandan, V.; Valliyammai, C.; Deepa, B.S. Switch Failure Detection in Software-Defined Networks. *Adv. Big Data Cloud Comput.* **2019**, *750*, 155–162.
48. Cascone, C.; Sanvito, D.; Pollini, L.; Capone, A.; Sansò, B. Fast failure detection and recovery in SDN with stateful data plane. *Int. J. Netw. Manag.* **2017**, *27*, e1957. [CrossRef]
49. Padma, V.; Yogesh, P. Proactive failure recovery in OpenFlow based Software Defined Networking. In Proceedings of the 3rd International Conference on Signal Processing, Communication and Networking (ICSCN), Chennai, India, 26–28 March 2015; pp. 1–6.
50. Huang, L.; Shen, Q.; Shao, W. Congestion Aware Fast Link Failure Recovery of SDN Network Based on Source Routing. *KSII Trans. Internet Inf. Syst.* **2017**, *11*, 5200–5222.

51. Capone, A.; Cascone, C.; Nguyen, A.Q.T.; Sanso, B. Detour planning for fast and reliable failure recovery in SDN with OpenState. In Proceedings of the International Conference on Design of Reliable Communication Networks, Kansas City, MO, USA, 24–27 March 2015; pp. 25–32.
52. Ramos, R.M.; Martinello, M.; Rothenberg, C.E. SlickFlow: Resilient source routing in Data Center Networks unlocked by OpenFlow. In Proceedings of the 38th Annual IEEE Conference on Local Computer Networks, Sydney, Australia, 21–24 October 2013; pp. 606–613.
53. Siminesh, C.N.; Grace, M.K.E.; Ranjitha, K. A proactive flow admission and re-routing scheme for load balancing and mitigation of congestion propagation in sdn data plane. *Int. J. Comput. Netw. Commun.* **2019**, *10*, 2018.
54. Soliman, M.; Nandy, B.; Lambadaris, I.; Ashwood-Smith, P. Exploring source routed forwarding in SDN-based WANs. In Proceedings of the 2014 IEEE International Conference on Communications, Sydney, Australia, 10–14 June 2014; pp. 3070–3075.
55. Soliman, M.; Nandy, B.; Lambadaris, I.; Ashwood-Smith, P. Source routed forwarding with software defined control, considerations and implications. In Proceedings of the 2012 ACM Conference on CONEXT Student Workshop, Nice, France, 10 December 2012; pp. 43–44.
56. Sangeetha Abdu, J.; Dong, M.; Godfrey, P.B. Towards a flexible data center fabric with source routing. In Proceedings of the 2015 1st ACM SIGCOMM Symposium on Software Defined Networking Research, Santa Clara, CA, USA, 17–18 June 2015; pp. 1–8.
57. Kreutz, D.; Ramos, F.M.V.; Veríssimo, P.E.; Rothenberg, C.E.; Azodolmolky, S.; Uhlig, S. Software-defined networking: A comprehensive survey. *Proc. IEEE* **2015**, *103*, 14–76. [CrossRef]
58. Chang, D.F.; Govindan, R.; Heidemann, J. The temporal and topological characteristics of bgp path changes in Network Protocols. In Proceedings of the 11th IEEE International Conference on Network Protocols, 2003, Proceedings, Atlanta, GA, USA, 4–7 November 2003; pp. 190–199.
59. Francois, P.; Bonaventure, O.; Decraene, B.; Coste, P.A. Avoiding disruptions during maintenance operations on bgp sessions. *IEEE Trans. Netw. Serv. Manag.* **2007**, *4*, 2007. [CrossRef]
60. Iannaccone, G.; Chuah, C.N.; Mortier, R.; Bhattacharyya, S.; Diot, C. Analysis of link failures in an ip backbone. In Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment, Marseille, France, 6–8 November 2002; ACM: New York, NY, USA, 2002; pp. 237–242.
61. Vissicchio, S.; Vanbever, L.; Pelsser, C.; Cittadini, L.; Francois, P.; Bonaventure, O. Improving network agility with seamless bgp reconfigurations. *IEEE/ACM Trans. Netw. (TON)* **2013**, *21*, 990–1002. [CrossRef]
62. Sharma, S.; Staessens, D.; Colle, D.; Pickavet, M.; Demeester, P. Enabling fast failure recovery in openflow networks. In Proceedings of the 2011 8th International Workshop on the Design of Reliable Communication Networks (DRCN), Kraków, Poland, 10–12 October 2011; pp. 164–171.
63. Muthumanikandan, V.; Valliyammai1, C. Link Failure Recovery Using Shortest Path Fast Rerouting Technique in SDN. *Wirel. Pers. Commun.* **2017**, *97*, 2475–2495. [CrossRef]
64. Astaneh, S.A.; Heydari, S.S. Optimization of SDN_ow operations in multi-failure restoration scenarios. *IEEE Trans. Netw. Serv. Manag.* **2016**, *13*, 421–432. [CrossRef]
65. Astaneh, S.; Heydari, S.S. Multi-failure restoration with minimal flow operations in software defined networks. In Proceedings of the 2015 11th International Conference on the Design of Reliable Communication Networks (DRCN), Kansas City, MO, USA, 24–27 March 2015; pp. 263–266.
66. Jin, X.; Liu, H.H.; Gandhi, R.; Kandula, S.; Mahajan, R.; Zhang, M.; Rexford, J.; Wattenhofer, R. Dynamic scheduling of network updates. *ACM SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 539–550. [CrossRef]
67. Malik, A.; Aziz, B.; Adda, M.; Ke, C.H. Optimisation Methods for Fast Restoration of Software-Defined Networks. *IEEE Access* **2017**, *5*, 16111–16123. [CrossRef]
68. Maesschalck, S.D.; Colle, D.; Lievens, I.; Pickavet, M.; Demeester, P.; Muaz, C.; Jaeger, M.; Inkret, R.; Mikac, B.; Derkacz, J. Pan-European optical transport networks: An availability-based comparison. *Photon. Netw. Commun.* **2003**, *5*, 203–225. [CrossRef]
69. Voellmy, A.; Wang, J. Scalable software defined network controllers. In Proceedings of the 2012 ACM SIGCOMM Conference, Helsinki, Finland, 13–17 August 2012; ACM: New York, NY, USA, 2012; pp. 289–290.
70. Karakus, M.; Durresi, A. A Survey: Control Plane Scalability Issues and Approaches in Software-Defined Networking (SDN). *Comput. Netw.* **2017**, *112*, 279–293. [CrossRef]
71. Qiu, K.; Yuan, J.; Zhao, J.; Wang, X.; Secci, S.; Fu, X. Efficient Recovery Path Computation for Fast Reroute in Large-scale Software Defined Networks. *IEEE J. Sel. Areas Commun.* **2018**, *37*, 1755–1768. [CrossRef]

72. Abdelaziz, A.; Fong, A.T.; Gani, A.; Garba, U.; Khan, S.; Akhunzada, A.; Talebian, H.; Choo, K.W.R. Distributed controller clustering in software defined networks. *PLoS ONE* **2017**. [CrossRef] [PubMed]
73. Sinha, Y.; Haribabu, K. A survey: Hybrid SDN. *J. Netw. Comput. Appl.* **2017**, *100*, 35–55.
74. Caesar, M.; Caldwell, D.; Feamster, N.; Rexford, J.; Shaikh, A.; van der Merwe, J. Design and implementation of a routing control platform. In Proceedings of the 2nd Conference on Symposium on Networked Systems Design&Implementation-Volume 2, Boston, MA, USA, 2 May 2005; USENIX Association: Berkeley, CA, USA, 2005; pp. 15–28.
75. Jin, C.; Lumezanu, C.; Xu, Q.; Zhang, Z.L.; Jiang, G. Telekinesis: Controlling legacy switch routing with OpenFlow in hybrid networks. In Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research, Santa Clara, CA, USA, 17–18 June 2015; ACM: New York, NY, USA, 2015; p. 20.
76. Tilmans, O.; Vissicchio, S.; Vanbever, L.; Rexford, J. Fibbing in action: On-demand load-balancing for better video delivery. In Proceedings of the ACM SIGCOMM 2016 Conference, Florianopolis, Brazil, 22–26 August 2016; ACM: New York, NY, USA, 2016; pp. 619–620.
77. Chu, C.Y.; Xi, K.; Luo, M.; Chao, H.J. Congestion-Aware Single Link Failure Recovery in Hybrid SDN Networks. In Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM), Kowloon, Hong Kong, 26 April–1 May 2015.
78. Rothenberg, C.E.; Nascimento, M.R.; Salvador, M.R.; Correa, C.N.A.; Lucena, S.C.D.; Raszuk, R. Revisiting routing control platforms with the eyes and muscles of software-defined networking. In Proceedings of the first ACM SIGCOMM workshop on HotSDN, Helsinki, Finland, 13 August 2012; Association for Computing Machinery: New York, NY, USA, 2012.
79. Vissicchio, S.; Vanbever, L.; Cittadini, L.; Xie, G.; Bonaventure, O. Safe Updates of Hybrid SDN Networks. *IEEE/ACM Trans. Netw.* **2017**, *25*, 1649–1662. [CrossRef]
80. Kvalbein, A.; Hansen, A.F.; Cicic, T.; Gjessing, S.; Lysne, O. Fast ip network recovery using multiple routing configurations. In Proceedings of the IEEE INFOCOM, Barcelona, Spain, 23–29 April 2006.
81. Suchara, M.; Xu, D.; Doverspike, R.; Johnson, D.; Rexford, J. Network architecture for joint failure recovery and traffic engineering. *ACM SIGMETRICS Perform. Eval. Rev.* **2011**, *39*, 97–108.
82. Reitblatt, M.; Canini, M.; Guha, A.; Foster, N. Fattire: Declarative fault tolerance for software-defined networks. In Proceedings of the second ACM SIGCOMM Workshop on HotSDN, Hong Kong, China, August 2013 ; Association for Computing Machinery: New York, NY, USA, 2013.
83. Vissicchio, S.; Vanbever, L.; Bonaventure, O. Opportunities and research challenges of hybrid software defined networks. *ACM SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 70–75. [CrossRef]
84. Vissicchio, S.; Vanbever, L.; Cittadini, L.; Xie, G.G.; Bonaventure, O. Safe routing reconfigurations with route redistribution. In Proceedings of the IEEE INFOCOM, Toronto, ON, Canada, 27 April–2 May 2014; pp. 199–207.
85. Chun-xiu, L.; Xin, L.; Ke, L.; Hong, Z.; Yu-long, S.; Shan-zhi, C. Toward software defined AS-level fast rerouting. *J. China Univ. Posts Telecommun.* **2014**, *21*, 100–108.
86. Rekhter, Y.; Li, T. *A Border Gateway Protocol 4 BGP-4*; ARPANET Working Group Requests for Comment; RFC-1771; DDN Network Information Center: Menlo Park, CA, USA, March 1995.
87. Skoldstrom, P.; Yedavalli, K. Network Virtualization and Resource Allocation in OpenFlow-Based Wide Area Networks. In Proceedings of the 2012 IEEE International Conference on Communications (ICC), Ottawa, ON, Canada, 10–15 June 2012.
88. OpenFlow Reference Switch Implementation. Available online: http://www.openflow.org/ (accessed on 27 January 2020).
89. Sharma, S.; Staessens, D.; Colle, D.; Pickavet, M.; Demeester, P. Fast failure recovery for in-band OpenFlow netwoks. In Proceedings of the 2013 9th International Conference on the Design of Reliable Communication Networks (DRCN), Budapest, Hungary, 4–7 March 2013.
90. Schiff, L.; Schmid, S.; Canini, M. Medieval: Towards A Self-Stabilizing Plug & Play In-Band SDN Control Network. In Proceedings of the ACM Sigcomm Symposium on SDN Research (SOSR), Santa Clara, CA, USA, 17–18 June 2015.
91. Sharma, S.; Staessens, D.; Colle, D.; Pickavet, M.; Demeester, P. In-Band Control Queuing and Failure Recovery Functionalities for OpenFlow. *IEEE Netw.* **2016**, *30*, 106–112. [CrossRef]
92. Schiff, L.; Kuznetsov, P.; Schmid, S. In-Band Synchronization for Distributed SDN Control Planes. *SIGCOMM Comput. Commun. Rev.* **2016**, *46*, 37–43. [CrossRef]

93. Evans, D. *The Internet of Things: How the Next Evolution of the Internet Is Changing Everything*; White Paper; Cisco: San Jose, CA, USA, 2011.
94. Xu, W.; Zhou, H.; Cheng, N.; Lyu, F.; Shi, W.; Chen, J.; Shen, X. Internet of Vehicles in Big Data Era. *IEEE/CAA J. Autom. Sin.* **2018**, *5*, 19–35. [CrossRef]
95. Wang, M.; Cui, Y.; Wang, X.; Xiao, S.; Jiang, J. Machine Learning for Networking: Workflow, Advances and Opportunities. *IEEE Netw.* **2018**, *32*, 92–99. [CrossRef]
96. Klaine, P.V.; Imran, M.A.; Onireti, O.; Souza, R.D. A survey of machine learning techniques applied to self-organizing cellular networks. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2392–2431. [CrossRef]
97. Khunteta, S.; Chavva, A.K.R. Deep Learning Based Link Failure Mitigation. In Proceedings of the 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Cancun, Mexico, 18–21 December 2017.
98. Srinivasan, S.M.; Truong-Huu, T.; Gurusamy, M. TE-Based Machine Learning Techniques for Link Fault Localization in Complex Networks. In Proceedings of the IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud), Barcelona, Spain, 6–8 August 2018.
99. Truong-Huu, T.; Mohan, P.M. Fast and Adaptive Failure Recovery using Machine Learning in Software Defined Networks. In Proceedings of the 2019 IEEE International Conference on Communications Workshops (ICC Workshops), Shanghai, China, 20–24 May 2019.
100. Dilmaghani, R.; Bottin, T. Evaluation of Software-Defined Networking for Mission Operations. In Proceedings of the 21st International Command and Control Research and Technology Symposium (ICCRTS), London, UK, 6–8 September 2016.
101. Du, P.; Pang, E.; Braun, T.; Gerla, M.; Hoffmannn, C.; Kim, J. Traffic Optimization in Software Defined Naval Network for Satellite Communications. In Proceedings of the IEEE Military Communications Conference (MILCOM), Baltimore, MD, USA, 23–25 October 2017.
102. Lantz, B.; Heller, B.; McKeown, N. A Network in a Laptop: Rapid Prototyping for Software-Defined Networks. In Proceedings of the 9th ACM Workshop on Hot Topics in Networks, Monterey, CA, USA, 20–21 October 2010.
103. Botta, A.; Dainotti, A.; Pescape, A. A Tool for the Generation of Realistic Network Workload for Emerging Networking Scenarios. In *Computer Networks*; Elsevier: Amsterdam, The Netherlands, 2012; Volume 56, pp. 3531–3547.
104. Huang, W.Y.; Chou, T.Y.; Hu, J.W.; Liu, T.L. Automatical end to end topology discovery and ow viewer on SDN. In Proceedings of the 2014 28th International Conference on Advanced Information Networking and Applications Workshops (WAINA), Victoria, BC, Canada, 13–16 May 2014; pp. 910–915.
105. Iperf. Available online: https://iperf.fr/ (accessed on 23 March 2020).
106. Lebiednik, B.; Mangal, A.; Tiwari, N. A survey and evaluation of data center network topologies. *arXiv* **2016**, arXiv:1605.01701.