Implementation Failure Recovery Mechanism using VLAN ID in Software Defined Networks

Heru Nurwarsito[1], Galih Prasetyo[2]
Faculty of Computer Science, University of Brawijaya, Malang, Indonesia[1, 2]

***Abstract*—Link failure is a common problem that occurs in software-defined networks. The most proposed approach for failure recovery is to use pre-configured backup paths in the switch. However, it may increase the number of traffic packets after the traffic is rerouted through the backup path. In this research, the proposed method is the implementation of a failure recovery mechanism by utilizing the fast failover group feature in OpenFlow to store pre-configured backup paths in the switch. The disrupted traffic packets will be labeled using the VLAN ID, which can be used as a matching field. Due to this capability, VLAN ID can aggregate traffic packets into one entry table as a match field in the forwarding rules. Through implementation and evaluation, it is shown that the system can build a backup path in the switch and reroute the disrupted traffic to the backup path. Based on the parameters used, the results show that the proposed approach achieves a recovery time of around 1.02-1.26ms. Additionally, it can reduce the number of traffic packets and has a low amount of packet loss compared to previous methods.**

Keywords*—Software-defined networks; openflow; link failure; failure recovery; VLAN ID; fast failover

I. Introduction

Software-defined networking is a new paradigm that changes the current network infrastructure. The SDN concept is to break down the network infrastructure by combining the control logic (control plane) of routers and switches that forward traffic (data plane) [1]. Unlike the conventional network concept, where the control plane and forwarding are tied directly to the same network device. This causes the network administrator to have to configure the device manually. Another disadvantage of conventional networks is that when a device encounters a problem, the network administrator must fix the problem directly on the device.

In SDN, several problems can occur when the routing process is executed, one of which is a link failure. Link failure is one of the problems on the network that causes delays and even packet loss so the throughput value decreases. Link failures consist of direct or indirect failures. In the case of direct failures, the switches detect the failure immediately and recover quickly, whereas, in the case of indirect failures, the link failure is not detected by the respective switch despite traffic overhead. Unidirectional link failures disrupt traffic and create a loop in the switch topology. Multiple link failures reduce network reliability performance [2].

Based on previous research that used proactive SDN methods to solve the link failure problem. When a link fails, a predetermined backup path is created and used in this proactive method. The switch closest to the link failure point will then reroute through the backup path to reach the destination [3]. Then in another research, using the rerouting method to overcome link failure was examined. When a link fails, high-priority packets are temporarily diverted to an alternate path, and then the packet is sent to the destination using the re-routing method to find the shortest path to the destination [4].

In the several methods previously mentioned, these methods can overcome link failures without involving the controller, thereby reducing failover time on SDN. However, these methods have several downsides, such as the need for a large number of flow entries to build backup lines and complex processes to maintain the lines alive. The backup path that has been created is then difficult to modify or adapt to changing network conditions, allowing for the possibility of congestion during the rerouting procedure.

Based on the problems described above and previous research, a test simulation will be developed to implement failure recovery on SDN by incorporating the fast failover group and VLAN ID features. This method works by creating a backup path for each link using the SDN architecture's fast failover group feature. Next, enable the VLAN ID feature, which is used as a match field in forwarding rules, to reduce traffic when packets are diverted to the backup path. This approach allows the system to recover from failures without involving the controller, reducing recovery time and minimizing the use of flow table memory on switches during link failures.

The rest of the paper is organized as follows: Section II provides a brief review of the literature relevant to our work and describes the main concepts around our approach. Section III explains the methodology of our approach is presented. Section IV explains the results and findings of the research. Section V is the discussion that presents the comparison result and findings with the previous studies. In Section VI, the conclusion of this research.

II. Related Work

A. Background

The Group Table-based Rerouting (GTR) method is one of the approaches used to find responses to single link failures through the fast failover (FF) group [5]. In general, a backup path is created for each link between the source and the destination; however, the backup path proposed in this research is created by calculating the most efficient path between adjacent switches. The controller periodically updates the lookup table on the controller and the FF group table on the

switch to determine what changes have occurred in the topology and network traffic. The research proposed a protection scheme for source routing-based backup links [6]. The proposed method controls packets going to the backup link by updating the source routing header. With SDN source routing, it can use VLAN tags to store packet routes in the source routing headers instead of being stored in switch flow entries. By using this method, it can reduce the number of flow entries needed to build a backup link. However, these approaches have limitations in the implementation process, which is quite complex to overcome existing problems in the fast failover mechanism.

Furthermore, in the research conducted [7], flow-based network management was proposed that can be programmed on the OpenFlow network. Researchers propose a method called Path Monitoring (PathMon), which encodes flow and path information as tags that can work flexibly. This method is implemented on a switch and uses OpenFlow 1.3, which supports VLAN tags to encode flow and path information as flexible tags so that the statistical data obtained to monitor network traffic on OpenFlow is more specific. The purpose of using VLAN IDs in this research is to make it easier for network administrators to monitor network traffic and collect the different statistical information needed. In further research, there are problems with the data center caused by the detection of elephant flow, which resulted in high network latency [8]. The proposed method is to use multipath routing, which can break down elephant flows into several mice flows that are distributed evenly on the network without detecting elephant flows. In this research, VLAN-based routing is used to reduce flow entry consumption, where the controller can instruct the switch to enter the path ID in the VLAN ID field in each packet from the flow when routing to the switch. The results showed a 32% reduction in flow entry on switches compared to the method without using VLANs. This research is distinct from one another in that researchers raise different issues. In the proposed approach, the problem to be solved is the use of fast failover, which requires a large number of flow entries when a link failure occurs.

B. Software Defined Networking

SDN provides a new approach to managing complex end-to-end connectivity and knowing the big picture of a network. A centralized network at the control layer allows management, configuration, security, and network resources to be optimized flexibly, dynamically, and automatically on SDN. It can be used for a variety of purposes, including control manipulation and network management, network virtualization, and providing a platform for building fast services [9].

As shown in Fig. 1, there is an SDN architecture consisting of an infrastructure layer related to the data plane that is in charge of forwarding. In the control layer, there is a component where the SDN controller is located. The application layer functions to make rules for the network. The control layer and the application layer are connected by the northbound API, while the infrastructure layer and the control layer are connected by the southbound API. In the southbound API, there is an SDN protocol, which is known as OpenFlow.

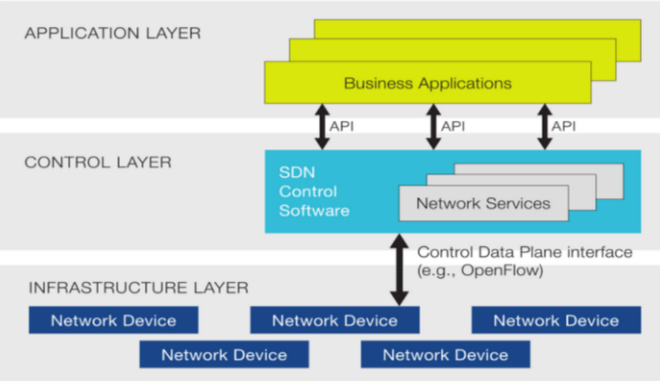

Fig. 1. SDN architecture

C. OpenFlow

OpenFlow is a standard protocol used in software-defined networks. This protocol is used for communication between the control plane and the data plane. The SDN controller can manage a collection of switches to manage network traffic. The controller communicates with the OpenFlow switch and manages the switch via the OpenFlow protocol [10]. Fig. 2 shows the OpenFlow Controller and the OpenFlow Switch are the two most important components of OpenFlow. The OpenFlow Controller manages the performance of the Switch by controlling paths and flows. Then, the OpenFlow Switch is part of the data plane, which functions to process data such as forwarding packets.

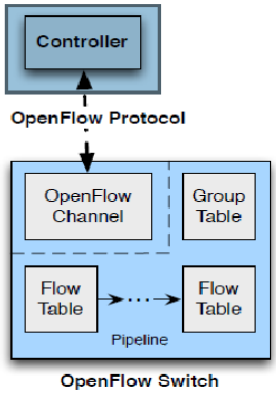

Fig. 2. Components of OpenFlow

In the OpenFlow protocol, each flow table on the switch has a flow entry, where each flow entry has a match field, a set of instructions to be applied to matching packets [11]. There are three main components in OpenFlow: the first is the table, which contains the flow table, meter table, and group table. The second is a secure channel that contains an SSL channel that is between the switch and the SDN controller. The third component is the OF protocol, which is used to control and manage switches [12].

D. Failure Recovery

Failure recovery is a network process that allows packets or flows that have experienced link failure to be recovered and forwarded to their destination. In the failure recovery process, there are two mechanisms: reactive mechanisms and proactive

mechanisms [13]. In a reactive mechanism, there is no backup path configured in the forwarding plane, so the controller immediately computes an alternative path after receiving a link failure notification message from the switch. Whereas in a proactive mechanism, there are two separate paths (the primary path and the backup path) that are configured by the controller in the forwarding plane before link failure occurs on the network. The fast failover feature is used to implement a backup path on a proactive mechanism. Fast failover is the ability of a flow table to create a group table that provides various ways of forwarding (primary and backup links) [14]. With this capability, fast failover can redirect disrupted flows to a backup link that has been configured in the flow table. Fault recovery is performed directly by the OpenFlow switch without involving the controller. In addition, failure recovery can be combined with backup path calculations, which are proactively installed by the controller.

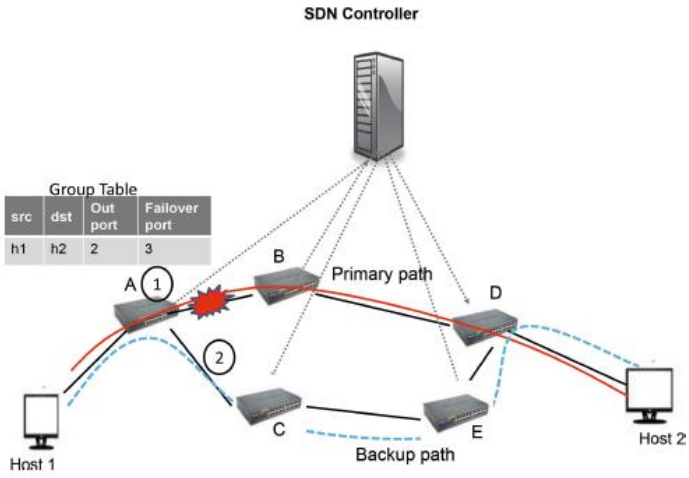

Fig. 3. Failure recovery mechanisms

Fig. 3 is an example of how the failure recovery mechanism works. There are 5 switches connected to 1 controller, and there are 2 hosts. In this topology, there is a primary path in switch A-B-D and a backup path in switch A-C-E-D. When there is a link failure on link A-B, packets from host 1 will be diverted to the backup path that has been configured in the flow table to switch A-C-E-D, so that packets can be sent to host 2.

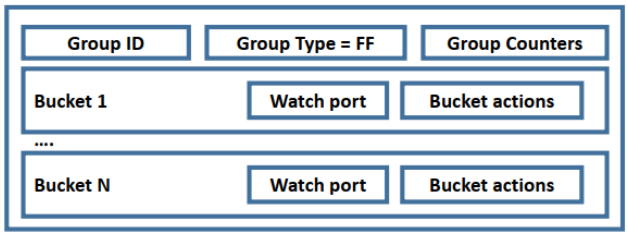

Fig. 4. Components of fast failover group

Fig. 4 shows the fast failover group component that is run by the SDN controller. Fast failover is a feature that reconfigures the link in the event of a failure. This feature utilizes OpenFlow 1.3 to run a group table that contains watch ports and action buckets that can monitor and act as long as the port status changes [15]. The SDN controller configures the switch with a flow table that can help the network recover when a link failure occurs. The flow table contains fast failover group rules that implement a path-switching mechanism if a link is down.

E. VLAN ID

VLAN-ID is one of the newest features introduced to OpenFlow in version 1.3. The VLAN mechanism can logically divide networks that are grouped based on VLAN ID. With this capability, VLANs can limit broadcast traffic on the network because they can only send packets to hosts that have the same VLAN ID. IEEE 802.1Q is the standard definition of VLANs. A VLAN tag contains 12 bits in the ethernet frame, so there can be up to 4,096 VLANs on a LAN. Implementation of VLAN tagging on the Ethernet protocol can create different broadcast sub-domains on the same LAN by including a VLAN number or tag for each subnet interface on the same switch [16].

In the OpenFlow protocol, in the flow table, there are flow entries that determine how a flow is processed and forwarded. Inside the flow entry, there are matching fields, actions, and counters. The matching field is used to match incoming packets. The action contains a set of instructions that are used to forward packets in various ways, for example, forwarding to a group table, one of which is the fast failover group. Then the counter is used to collect statistics on a particular flow, for example, the number of packets that have been received, the number of bytes, and the duration of the flow.

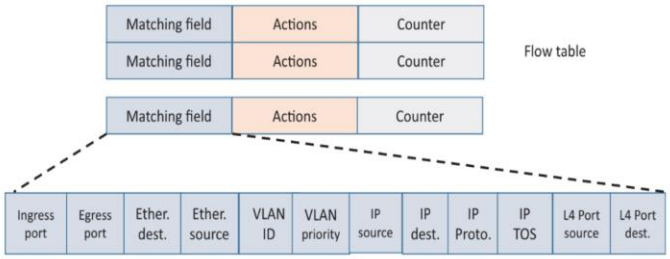

Fig. 5. Fields in the OpenFlow protocol

As shown in Fig. 5, there are 12 match fields, which are collectively referred to as the "basic twelve-tuple of match fields". Flow entries are processed sequentially, and when a match is found, the matching process against the flow table will be stopped [17]. In addition, the flow table is also equipped with a frame/byte counter that provides an indication of flow statistics on all ports so that the controller knows all the conditions that occur in the network [18]. Several actions can be performed by the OpenFlow protocol, such as sending packets to several ports, adding, removing, or modifying a VLAN tag, deleting packets, or sending packets to the controller.

III. PROPOSED APPROACH

In OpenFlow, a specific flow can be defined as a collection of matching fields. Therefore, VLAN ID can be used as a flow ID, which can be forwarded based on flow entry. The use of VLAN ID can reduce interference with route flow and thus

reduce switch memory consumption. The failure recovery mechanism proposed in this research uses the VLAN ID feature on OpenFlow to collect failed flows. Each switch and a link is associated with a VLAN ID that can redirect flow to a backup path configured by fast failover on the switch.

Based on the topology that will be used in this research, the failure recovery mechanism will be implemented using OpenFlow 1.3. This mechanism provides a primary path for forwarding and a backup path for diverting packets to an alternative path when a link failure occurs in the primary path. Based on Fig. 6, shows the failure recovery mechanism in the topology. In this topology, there is a primary path in S1-S2-S3-S4, while the backup path is in S2-S5-S6-S7-S4. Then, with the backup path configuration that has been made in S2, traffic can be diverted from port 3 to S5-S6-S7-S4 without making a round trip to the controller, so that packets can be sent to the destination.

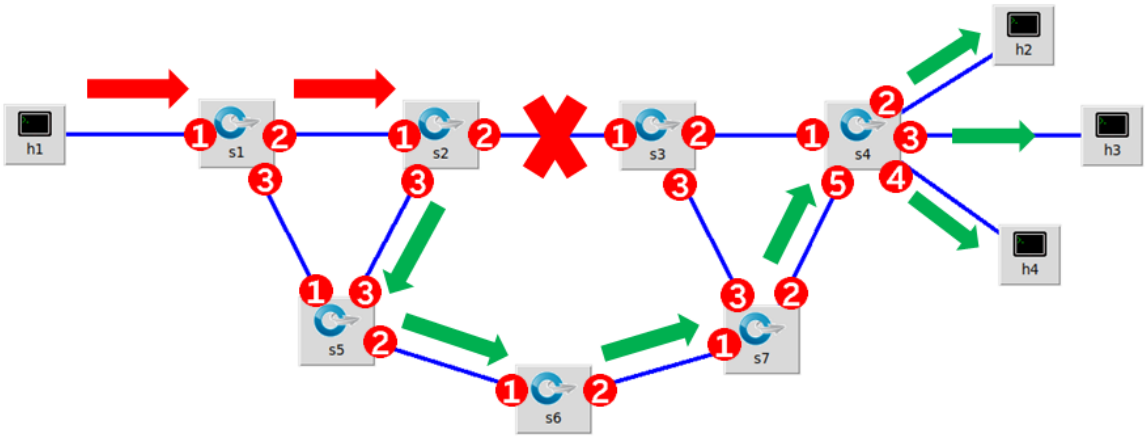

Fig. 6. Experimental topology

In the network topology, there is a link failure in S2-S3, so packets cannot pass through the link. Furthermore, there is H1, which will send packets to H2, H3, and H4, so there are three traffic flows in the topology. When the S2-S3 link fails, the controller will update the network topology by removing the failed link. In flow table S2, traffic is forwarded to group table 2 with the fast failover type, which is sent to output port 3.

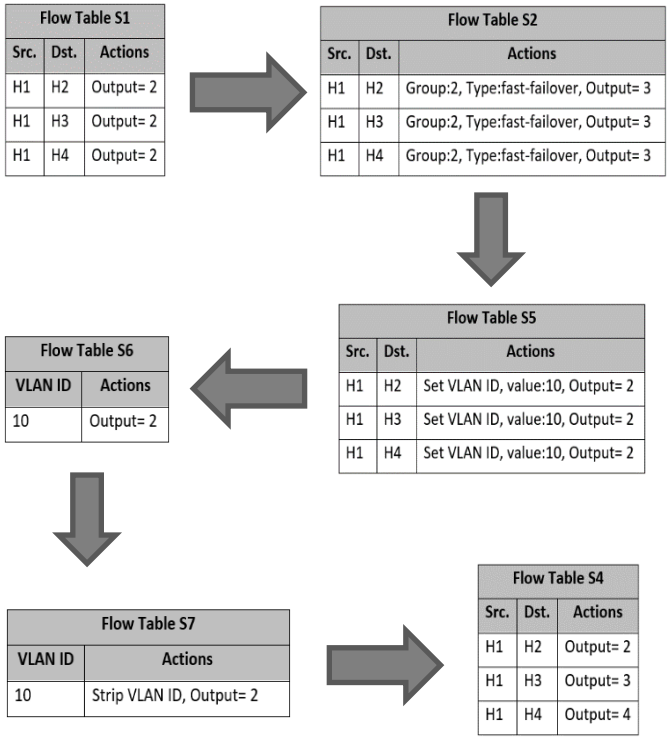

Fig. 7. Labelling process of VLAN ID

In Fig. 7, when the packet arrives at S5, configure the VLAN ID in the access port by accepting all packets that do not have a VLAN header. Then add a VLAN ID tag with a value of 10 for each incoming packet in S5. Then, the value of the VLAN ID is used as a match field in the switch connected to S5 and S6 via port 2. When it arrives at the next switch, packets will become one flow with a match field VLAN ID of 10. In S6, there is a packet with a VLAN ID as a match field with a value of 10 that has been configured on the previous switch. When a packet with a VLAN ID matches the flow match field, the packet can be forwarded based on the action specified in the flow table. Then, when the flow arrives at S7, there is an action with the Strip VLAN ID that functions to delete the value from the VLAN ID. The process is in the output access port, so the VLAN header has been deleted when it goes to the output port. Furthermore, when the flow arrives at S4, it will be returned to three traffic flows. Thus, in flow table S4, the three traffic flows can be forwarded according to the forwarding rules in S4 without requiring changes to the flow table S4.

IV. Performance Evaluation

A. Testbed Configuration

The topology shown in Fig. 6 was implemented on Mininet as the network emulator and select Ryu as the controller. Mininet supports different types of switches. In this case, we used OpenVSwitch to support the fast failover group and VLAN ID features in OpenFlow 1.3. Furthermore, because our experiment was carried out in a controlled environment, we used OpenVSwitch to install the flows directly in each switch of the network topology using the script-line program *ovs-ofctl*. To collect statistics and monitor the behavior of TCP traffic generated by the IPERF application. The main characteristics

of the laptop on which the tests were conducted are as follows: Processor: Intel Core i7-8550U, CPU running at 1.99 GHz; RAM: 16 GB; operating system: Linux Ubuntu 20.04 LTS 64-bit on a VMware workstation.

B. Recovery Time

The recovery time evaluation is carried out to find out how long it takes for packets to be diverted to the backup path when a link fails. The analysis compares the recovery time in failure recovery with VLAN ID and fast failover.

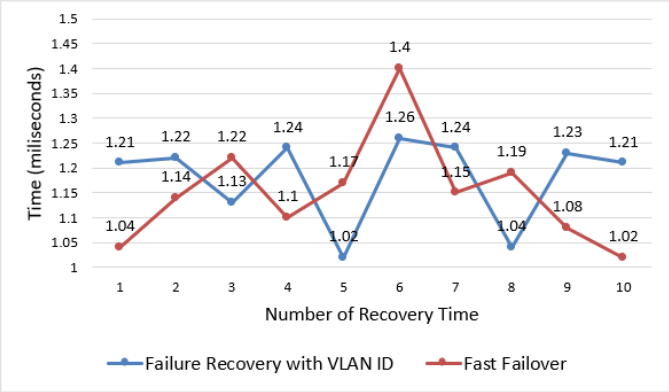

Fig. 8. Failure recovery time

In Fig. 8, there is a graph of the recovery time test. Based on the test results, the minimum time required for failure recovery with a VLAN ID is 1.02 ms, while the maximum time is 1.26ms. As a result, the time required to perform recovery in this mechanism is 1.02-1.26ms. Whereas for fast failover, the minimum time is 1.02ms, and has the maximum time of 1.4ms. As a result, when a link fails, recovery takes 1.02-1.4 ms on fast failover. Based on the results obtained in this test, the recovery time required in the failure recovery mechanism with VLAN ID is smaller than that required in the fast failover mechanism.

C. Traffic Packets

The Traffic packet evaluation is used to find out how much packet traffic is transmitted to the destination. The analysis of this evaluation is used to determine the performance of the VLAN ID as a matching field in sending packets to the destination.

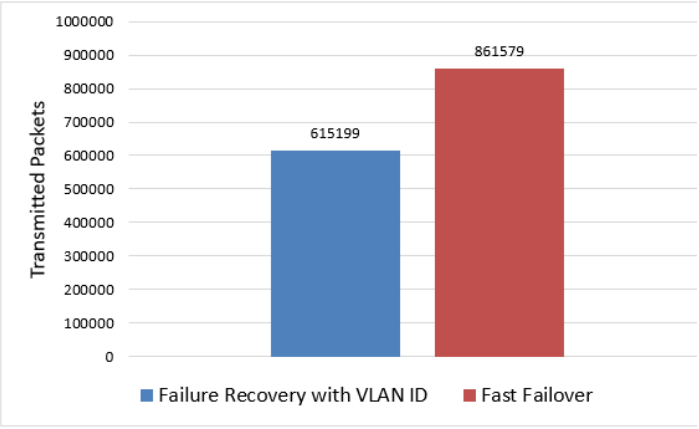

Fig. 9. Total number of traffic packets

As shown in Fig. 9, there is a traffic packet evaluation. Based on the two mechanisms tested, the failure recovery mechanism with VLAN ID resulted in a total of 615,199 traffic packets. Meanwhile, in failure recovery, the number of packets generated in this test was 861,579 packets. According to the results obtained from the test, failure recovery with VLAN ID produces less packet traffic than the fast failover mechanism.

D. Packet Loss

Packet loss evaluation is carried out to find out how many packets are lost when sending packets from host 1 to host 2 when a link failure occurs. The duration of each test to be carried out is 10 seconds, and the test is carried out five times. The test will be carried out with a different total number of streams.

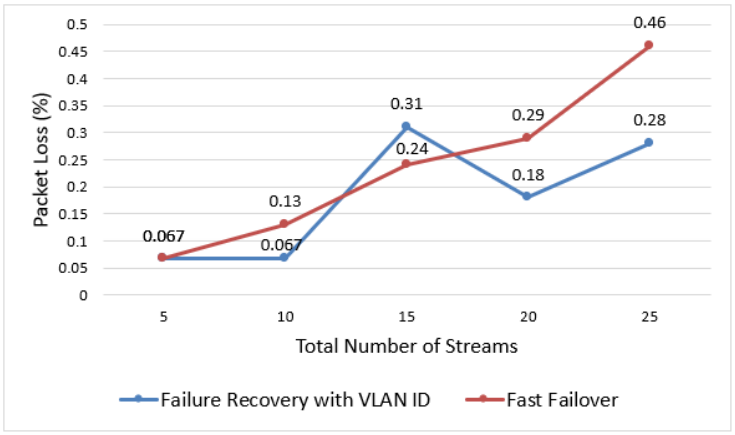

Fig. 10. Packet loss rate

In Fig. 10, there is a graph of the results of the packet loss evaluation that has been done. In the first test using 5 streams, the failure recovery mechanism with VLAN ID has a packet loss value of 0.067%, and this value increases when the last test uses 25 streams at 0.28%. Whereas in fast failover, in the first test, it has the same packet loss value of 0.067%. But in the last test using 25 streams, the packet loss value was 0.46%. Based on the results obtained in this test, the smallest packet loss value for the two mechanisms is 0.067%. Whereas in the last test, the failure recovery mechanism with VLAN ID had a smaller packet loss value compared to fast failover.

V. Discussion

In the recovery time test results, the failure recovery mechanism with VLAN ID shows results of 1.02-1.26ms with an average yield of 1.18ms to perform recovery after a link failure occurs. In research conducted [19], the time needed to detect and recover a single link failure is at least around 10-20 ms. Then, research [20], states that the need to detect and perform recovery on operator-scale networks must be carried out in 50ms time intervals. The results of the packet loss test that has been carried out show that the average value of packet loss in the failure recovery mechanism with a VLAN ID is 0.18%, with the highest packet loss being 0.28%. The results of the packet loss test are still considered good, based on research conducted [21] which states that packet loss with a ratio of 5-10% can affect network quality. Whereas in audio and video

stream scenarios, the range of acceptable packet loss is between 1 and 2.5%.

VI. CONCLUSION

Based on the results of the analysis of this research it can be concluded that compared to the fast failover method used in SDN, it mainly has three advantages: First, in the recovery time test results, the failure recovery mechanism with VLAN ID shows results of 1.02-1.26ms to perform recovery after a link failure occurs. Whereas in the fast failover mechanism, the time needed to perform recovery is 1.02-1.4ms. Second, it shows that the use of VLAN ID in failure recovery is proven to be able to reduce the amount of traffic packet when a link failure occurs. Third, the results of the packet loss evaluation that has been carried out show that the average value of packet loss in the failure recovery mechanism with a VLAN ID is 0.18%, with the highest packet loss is 0.28%. Based on the evaluation results, our proposed approach has better results than the fast failover method. However, the major drawback of our proposed approach is that the mechanism is less dynamic because we implement fast failover groups and VLAN IDs directly in the switch. Perhaps we can present a solution to the problem and provide direction for our future work.

REFERENCES

[1] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," Proceedings of the IEEE, vol. 103, no. 1, pp. 14–76, Jan. 2015, doi: 10.1109/JPROC.2014.2371999.

[2] V. Muthumanikandan and C. Valliyammai, "A survey on link failures in software defined networks," ICoAC 2015 - 7th International Conference on Advanced Computing, Sep. 2016, doi: 10.1109/ICOAC.2015.7562808.

[3] R. Kanagavelu and Y. Zhu, "A pro-active and adaptive mechanism for fast failure recovery in SDN data centers," Advances in Intelligent Systems and Computing, vol. 886, pp. 239–257, 2019, doi: 10.1007/978-3-030-03402-3_17/COVER.

[4] V. Muthumanikandan and C. Valliyammai, "Link Failure Recovery Using Shortest Path Fast Rerouting Technique in SDN," Wireless Personal Communications 2017 97:2, vol. 97, no. 2, pp. 2475–2495, Jun. 2017, doi: 10.1007/S11277-017-4618-0.

[5] S. Petale and J. Thangaraj, "Link Failure Recovery Mechanism in Software Defined Networks," IEEE Journal on Selected Areas in Communications, vol. 38, no. 7, pp. 1285–1292, Jul. 2020, doi: 10.1109/JSAC.2020.2986668.

[6] L. Huang, Q. Shen, and W. Shao, "A source routing based link protection method for link failure in SDN," 2016 2nd IEEE International Conference on Computer and Communications, ICCC 2016 - Proceedings, pp. 2588–2594, May 2017, doi: 10.1109/COMPCOMM.2016.7925166.

[7] M. H. Wang, S. Y. Wu, L. H. Yen, and C. C. Tseng, "PathMon: Path-specific traffic monitoring in OpenFlow-enabled networks," International Conference on Ubiquitous and Future Networks, ICUFN, vol. 2016-August, pp. 775–780, Aug. 2016, doi: 10.1109/ICUFN.2016.7537143.

[8] S. Chakraborty and C. Chen, "A low-latency multipath routing without elephant flow detection for data centers," IEEE International Conference on High Performance Switching and Routing, HPSR, vol. 2016-July, pp. 49–54, Jul. 2016, doi: 10.1109/HPSR.2016.7525638.

[9] C. Decusatis et al., "Dynamic, software-defined service provider network infrastructure and cloud drivers for SDN adoption," 2013 IEEE International Conference on Communications Workshops, ICC 2013, pp. 235–239, 2013, doi: 10.1109/ICCW.2013.6649235.

[10] F. Hu, Q. Hao, and K. Bao, "A survey on software-defined network and OpenFlow: From concept to implementation," IEEE Communications Surveys and Tutorials, vol. 16, no. 4, pp. 2181–2206, Apr. 2014, doi: 10.1109/COMST.2014.2326417.

[11] A. Mukherjee, R. A. Saeed, S. Dutta, and M. K. Naskar, "Fault tracking framework for software-defined networking (SDN)," Resource Allocation in Next-Generation Broadband Wireless Access Networks, pp. 247–272, Feb. 2017, doi: 10.4018/978-1-5225-2023-8.CH011.

[12] S. Jamali, A. Badirzadeh, and M. S. Siapoush, "On the use of the genetic programming for balanced load distribution in software-defined networks," Digital Communications and Networks, vol. 5, no. 4, pp. 288–296, Nov. 2019, doi: 10.1016/J.DCAN.2019.10.002.

[13] [13] R. Ahmed, E. Alfaki, and M. Nawari, "Fast failure detection and recovery mechanism for dynamic networks using software-defined networking," Proceedings of 2016 Conference of Basic Sciences and Engineering Studies, SGCAC 2016, pp. 167–170, Apr. 2016, doi: 10.1109/SGCAC.2016.7458023.

[14] E. Molina, E. Jacob, J. Matias, N. Moreira, and A. Astarloa, "Using Software Defined Networking to manage and control IEC 61850-based systems," Computers & Electrical Engineering, vol. 43, pp. 142–154, Apr. 2015, doi: 10.1016/J.COMPELECENG.2014.10.016.

[15] K. Halba, C. Mahmoudi, and E. Griffor, "Robust safety for autonomous vehicles through reconfigurable networking," Electronic Proceedings in Theoretical Computer Science, EPTCS, vol. 269, pp. 48–58, Apr. 2018, doi: 10.4204/EPTCS.269.5.

[16] M. B. Lehocine and M. Batouche, "Flexibility of managing VLAN filtering and segmentation in SDN networks," 2017 International Symposium on Networks, Computers and Communications, ISNCC 2017, Oct. 2017, doi: 10.1109/ISNCC.2017.8071999.

[17] P. Göransson, C. Black, and T. Culver, "The OpenFlow Specification," Software Defined Networks, pp. 89–136, Jan. 2017, doi: 10.1016/B978-0-12-804555-8.00005-3.

[18] G. Pujolle, "Software Networks: Virtualization, SDN, 5G, and Security," Wiley eBooks, 2020. https://ieeexplore.ieee.org/book/9116614 (accessed Oct. 03, 2022).

[19] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "Enabling fast failure recovery in OpenFlow networks," pp. 164–171, Dec. 2011, doi: 10.1109/DRCN.2011.6076899.

[20] D. Staessens, S. Sharma, D. Colle, M. Pickavet, and P. Demeester, "Software defined networking: Meeting carrier grade requirements," IEEE Workshop on Local and Metropolitan Area Networks, 2011, doi: 10.1109/LANMAN.2011.6076935.

[21] M. Pundir and J. K. Sandhu, "A Systematic Review of Quality of Service in Wireless Sensor Networks using Machine Learning: Recent Trend and Future Vision," Journal of Network and Computer Applications, vol. 188, p. 103084, Aug. 2021, doi: 10.1016/J.JNCA.2021.103084.