IJIST
P ISSN : 2618-1630
E ISSN : 2909-6130
Original Article

Reliability Awareness Multiple Path Installation in Software Defined Networking using Machine Learning Algorithm

Muzammal Majeed[1], Rashid Amin[2], Farrukh Shoukat Ali[1], Adeel Ahmed[3], Mudassar Hussain
[1] (Department of Computer Science, University of Engineering and Technology, Taxila).
[2] (Department of Computer Science, University of Chakwal, Pakistan).
[3] Department of Computer Science, Quaid-i-Azam University, Islamabad.
[4] (Department of Computer Science, University of Wah, Wah Cantt, Pakistan).
* **Correspondence:** Rashid Amin, **rashid4nw@gmail.com**

Citation |Majeed. M, Amin. R, Ali. S. F, Ahmed. A, Hussain. M, "Reliability Awareness Multiple Path Installation in Software Defined Networking using Machine Learning Algorithm," Int. J. Innov. Sci. Technol., Special Issue, pp 158-172, 2022.
Received | July 20, 2022; **Revised** | August 05, 2022; **Accepted** | August 19, 2022; **Published** | August 21, 2022.
DOI| https://doi.org/10.33411/IJIST/2022040510

Link failure is still a severe problem in today's networking system. Transmission delays and data packet loss cause link failure in the network. Rapid connection recovery after a link breakdown is an important topic in networking. The failure of the networking link must be recovered whenever possible because it could cause blockage of network traffic and obstruct normal network operation. To overcome this difficulty, backup or secondary channels can be chosen adaptively and proactively in SDN based on data traffic dynamics in the network. When a network connection fails, packets must find a different way to their destination. The goal of this research is to find an alternative way. Our proposed methodology uses a machine-learning algorithm called Linear Regression to uncover alternative network paths. To provide for speedy failure recovery, the controller communicates this alternate path to the network switches ahead of time. We train, test, and validate the learning model using a machine learning approach. To simulate our proposed technique and locate the trials, we use the Mininet network simulator. The simulation results show that our suggested approach recovers link failure most effectively compared to existing solutions.

Keywords: SDN, Link failure, Failure Recovery, Machine Learning, Linear Regression.

Introduction

There are various devices connected to the computer network, and it increases every day. The network's ability to transport faster packets from one network point to another becomes a major issue[1]. The network system's ability to swiftly transfer data packets has been hampered by node failure, connection failure, bandwidth limitations, and link congestion. In today's networking world, link failure is a serious problem and a source of trouble for network administrators[2]. If a network link fails, traffic that relied on it must be rerouted to another link to ensure that data is delivered reliably. The primary path in a networking system is described as the way through which actual network traffic passes, whereas the secondary path is the path used for network traffic in the event of a failure.

Every available node in the network calculates primary and secondary routes for every target computer in the network[3].

Traditional Network.

The data and control planes are available in the traditional network switches. As a result of this issue, the network consumes excessive bandwidth[4]. When information about network configuration overflows, a bottleneck occurs because the data and control planes share the network's bandwidth. The amount of data kept in the control plane grows when a network link fails. When a network link fails, all connected devices receive information from the nodes about the failure, producing network congestion[5].

Software Defined Networking.

In a software-defined network structure, both the data and control planes are separate from each other[6]. The network's logic and rules are transferred from network devices to a central device known as the controller. With a southbound application programming interface, this main network controller regulates data transfer over the network. The benefit of this centralized controller is that all new network services, applications, and functions may be configured flexibly with low operational costs and capital expenditures[7]. When a link fails in SDN, the node connected to the failed link sends a link failure notification to the main controller directly. The control panel then manages the data journey alternate path. Because the network's data plane is not disrupted, the computer network's competency increases. The way of communication between network switches and the main controller is the OpenFlow protocol[8], [9].

Machine Learning.

A type of data analysis called machine learning automates the process of developing an analytical model. This technology seeks to mimic human thought patterns and generate predictions based on massive data. These algorithms are widely employed in several applications and packages, including transportation, system performance, many software services, cloud computing, networking, and energy, thanks to advancements in data storage technology[10]. One of the most significant advantages of machine learning is its ability to solve even the most difficult situations[11][12].

Linear Regression.

It is one of the most popular and widely used models in the machine learning area. Using statistics performs the predictive analysis. Other continuous/real or quantitative values, such as sales, salaries, age, and product prices, are likewise predicted using this method. The process for linear regression reveals a linear relationship between one or more independent (x) variables and a dependent (y) variable[13]. Since linear regression establishes a linear relationship between them, the value of the dependent variable changes as the value of the independent variable changes. A sloping straight line is used to represent the relationship between the variables in the linear regression model[14].

Figure 1 describes 3 plains that are Control Plane, Data Plane, and Artificial Intelligence plane in SDN assembly.

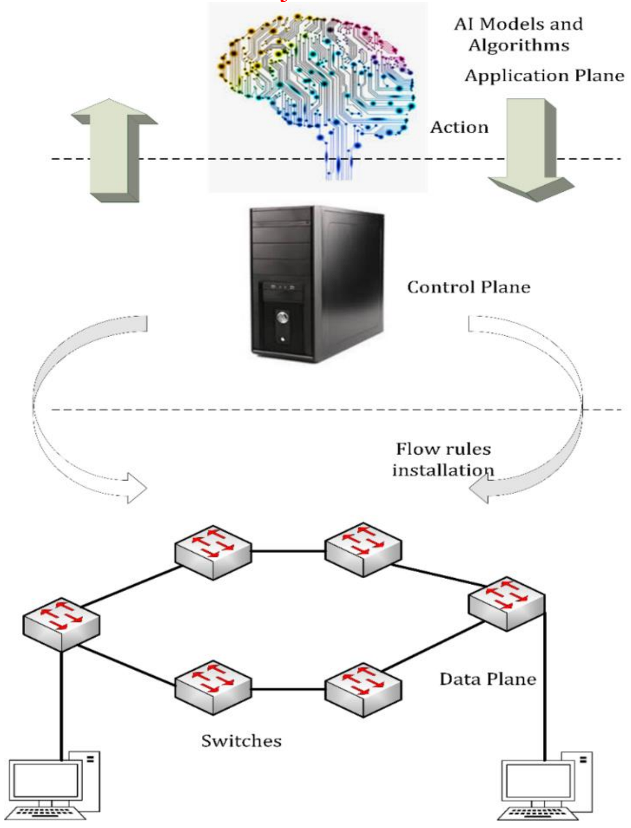

Figure 1 Three layers of SDN scenario

Research Objectives.

The research objectives of our proposed work are as follows:

1. Detection of faults in routes
2. Computation of alternatives routes
3. Validation of successful data transmission

Novelty Statement.

In our work, we discuss the problem of a connection failure in software-defined networking and propose a novel machine learning-based solution to mitigate the link failure issue[15]. The traditional approaches for link failure are not efficient to handle link failure as it involves the identification of available alternative links, and it also needs to compute the reliability of the links. So, we proposed a linear regression model to compute the reliability of alternative paths for smooth communication. The proposed scheme improves the latency rate, response time, and overhead of existing approaches.

Literature Review.

Malik et al. [16]say that in both fields of academic and industrial, Software Defined Networking (SDN) plays an important role nowadays. There are many benefits to the traditional networking system using SDN. In the modern age of the networking data plane, fault management is divided into two methods that are proactive and reactive[17]. Whenever any failure occurs or packet loss in the network, these recovery and fault management techniques are run. Due to convergence time, these strategies are used when the new network's routes must be permitted to forward the affected signals rather than dropping them completely. This type of convergence causes unavailability of signals and disruption of temporary service. In this article, researchers proposed a new system for data plane fault management in SDN. The purpose of this technique is to eradicate the process of convergence instead of accelerating the recovery and failure detection. The authors also defined a new framework named Smart Routing. In this framework, when the link failure occurs, the entire network controller gets the alarm about this failure and saves all the dangerous routes before the occurrence of failure. This proposed method aims to reduce the disturbance in the network service and increase network availability. Authors prove their method with a lot of experiments and describe how this inherent model executes and explains its impacts on improving network service availability.

Garoui et al. [18] discuss an environment called a smart city that connects a lot of devices like sensors, mobiles, cars, and actuators. In this environment, new real-time applications are provided by a system called Intelligent and Connected Transportation System (ICTS). Internet-of-Things (IoT) technologies are at the heart of the created packages, which carry modern summons like scalability and heterogeneity and need cutting-edge communication system solutions. The modern network routing protocols cannot acquire these restrictions because of field knowledge supported by a specific computer system demonstrating partial visibility of the network system. Due to the dynamic topology and variation of the network, developing routing systems to match the ever-changing network requirements is becoming increasingly complex. With the help of the software's characteristics, SDN delivers a fresh picture of the whole networking system and manages all the devices connected to the network. When using the SDN-based network, the major issue is minimizing the broadcast delay between all the computer systems connected to the network. To address this network limitation, other Machine Learning (ML) techniques as prediction solutions that are well-defined and realistic are required. To recover the delay that occurs in the network, the authors present novel routing rules or protocols methodology based on SDN and Naive Bays algorithms (ML algorithms)[19]. According to the simulation's results, their routing system exceeds the competition regarding the end-to-end delay and data transmission ratio.

Moazzeni et al.[20] explain in their research article the purpose of SDN is to control the very complex functionality of the controller connected to the networking system and make its scalability easier. The controlling system in the SDN manages and handled by a logically centralized device called a controller. When the link fails in the network first, this controller activates its functionality, and here data plane is unavoidable. This single controller is not so reliable in the network because of its single point of failure. The authors say that we can use distributed controllers to reduce this complexity in the network. In their proposed method, the researchers divided the big network into different small subnetworks, where a single controller manages each network to lessen the effects of failure. Every subnetwork's consistency is determined by considering the number of nodes in the network and examining the connection failure rate, after which it is distributed among the controllers using the Dijkstra Algorithm and Leader Election. With the newly established Coordinator Finder Algorithm, the controller with the maximum consistency rate is chosen as the manager[21]. When a controller in the network fails in the transitory technique, the coordinator picks a suitable controller for the smaller network., improving accuracy and fault tolerance while lowering latency. The authors use the Colored Petri-Net model to verify this proposed method.

For various years, the link failure problem has been a dangerous concern in the field of Software-Defined Networks (SDN), according to Hu et al[22]. To guarantee network strength and availability, both proactive and reactive methods are available. The latter requires a lot of time to recover from a fault, while the former wastes a lot of TCAM and bandwidth. The authors of this work introduce FTLink, an effective and adaptable link fault tolerance technique. The writer gathers network information to create various backup connections for every connection in the flow of the primary path. The author attempts to design network backup connections as a multi-objective optimization problem with switch TCAMs and link bandwidths set to a minimum. By combining flow features, the author creates a two-step heuristic algorithm to inspect the backup links set. Step 1 of the algorithm involves planning alternative links for each link that transmits elephant flows using the greedy tracing technique, and Step 2 entails employing the bidirectional searching technique to plan alternative links for each link that transmits mice flows. The resulting backup links and additional flow rules are then stored in a global matching table created by the controller programmers. This is how elastic fault recovery works: after confirming a broken connection, FTLink searches for appropriate table entries and provides links by adding additional flow rules to the switches. When compared to competing techniques, the results demonstrate that FTLink is successful in terms of high-efficiency TCAM and transmission capacity use, with a recovery time of fewer than 30 milliseconds. Experiments on models conducted in the ideal network environment provide more evidence of FTLink's capabilities.

It is an important problem in the field of the network to recover link failure with high speed and efficiency. To address this problem in SDN, multiple routes are defined in proactive and adaptive manners according to the dynamic condition of traffic. In [23] Huu et al. say that proactive techniques make use of only the network structure knowledge, and to compute the backup path individual can use static load. The authors proposed a machine learning system named Traffic Engineering (TE) that can understand and define the efficient route, learn the traffic dynamically, and make backup paths after link failure adaptively. Authors do experiments, train, test, and also validate the learning model using various machine learning tools like linear regression, decision tree, SVM, NN, gradient boosting, and random forest. The authors used the Mininet Evaluation Platform for their experiments. Their results show that by using their proposed method, failure recovery time is reduced by up to 50%, and network bandwidth utilization improves by 20% as compared to the baseline technique.

Problem Statement.

It can be observed that there would be a deterministic switch activation process sequence in the physical network topology as the result of the way the registration mechanism was designed, while the controller serves as a manager shown in Fig 2. Switch 1 was immediately connected to the main SDN controller in this sequence, and this switch received its initial response from the controller. It became a live switch and attempted to transmit messages to all of its neighbors. Later, all of the switches will be triggered. An activation wave is formed as a result of this operation, which propagates between all switches and activates all switches connected to the network. State machines within each switch explain this process, not the controller. In this network, the creation process controller performs the role of an information assembler by putting the logical topology representation together[24].

In figure 2 there are 6 switches named s1, s2, s3, s4 s5, and s6 connected, and these all switches are directly connected with a single main SDN controller. Five hostesses, PC 1 to PC 5 with IP address 192.168.1.101, 192.168.1.102, 192.168.1.103, 192.168.1.104, 192.168.1.105 simultaneously connected in this network for data communication with the switches. If any host wants to communicate with another host, it will communicate through s1, to s6 switches.

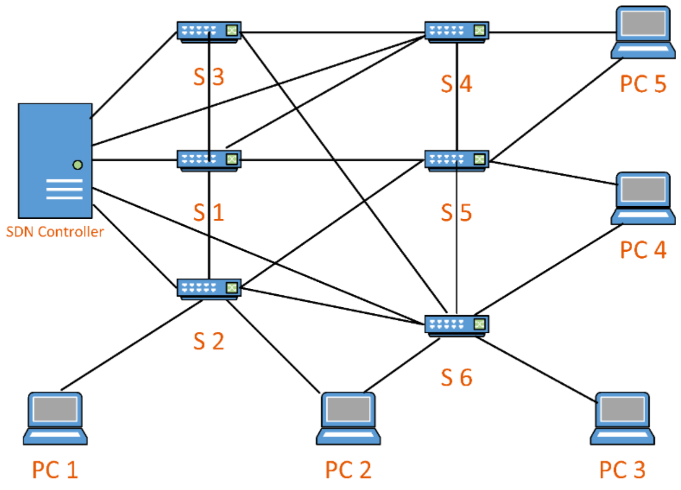

Figure 2 Network Architecture

If PC 1 wants to send a message to PC 5, it must first connect with switch 2, whose forwarding plane will send a command to the main SDN controller requesting or asking from which path the message should be sent to PC 5. The SDN controller is connected to all the network switches and has access to the network's data. There are several ways to get from PC 1 to PC 5. This controller examines all paths from PC 1 to PC 5 and attempts to determine the most efficient path between the two computers. When one or more links in the network break and one or more paths from source to destination are busy for whatever reason, the network has a problem. Suppose multiple data pathways become clogged or overburdened. To ensure smooth communication, the SDN controller now has difficulty determining the appropriate path with the lowest latency rate from source to destination. It is the job of the main SDN controller to maintain and construct another alternative link between sender and receiver for efficient communication whenever a particular link among the computer systems in the network breaks down.

To summarize, we're searching for a group solution in which all devices are pre-configured with backup forwarding data rules to address all of the network difficulties described above. Because those forwarding rules will be calculated for each link or node failure, the primary and backup paths will be as short as feasible, substantially improving the outdated path disjointed protection strategy.

Material and Methods.

We use the Linear Regression model for our proposed solution. It is one of the most popular and widely used models in the machine learning area. Using statistics performs the predictive analysis. Other continuous/real or quantitative values, such as sales, salaries, age, and product prices, are likewise predicted using this method. The process for linear regression reveals a linear relationship between one or more independent (x) variables and a dependent (y) variable. The value of the dependent variable changes as the value of the independent variable changes because linear regression establishes a linear relationship between them. A sloping straight line is used to represent the relationship between the variables in the linear regression model[25]. SDN network takes some kinds of information from network topologies like link failure and traffic data. If network bandwidth increases from its limits, it goes to link failure and congested link[26]. To solve this problem, the controller will request a linear regression model. The linear regression model now examines all network traffic and calculates new pathways and secure links between sender and destination. Our proposed strategy is depicted in Figure 3.

The controller in Software Defined Networking scans the entire networking system and assembles or prepares to receive flows from the sending end to the receiving end[27]. When the controller gets a route calculation request from a host computer delivering data via a networking node, random Quality of Services (QoS) matrices are generated for this type of flow to mimic all QoS needs. The path from sending to receiving computer or network node is estimated using algorithms placed in the main controller, and then it is checked to see if it fits all of the QoS standards[28]. If this happens, an SDN controller is used to program all of the network's machines; otherwise, the flow is stopped.

When a large amount of data must be transferred from PC 1 to PC 5, the controller's algorithm takes different paths, such as switching 4 to switch 1 and switching 4 to switch 2. When a large amount of traffic needs to be transferred from PC 3 to PC 5 or PC 4 to PC 5 at the same time, the network's pathways between PC 1 and PC 5 and PC 4 to PC 5 become clogged. As a result, there is an increase in bandwidth overhead, and these pathways become congested.

We set a 90 percent threshold link value for network congestion prevention in our suggested effort. When network traffic exceeds the link threshold value, the controller instructs the linear regression model to find new multiple alternate paths to bypass network congestion. For data traffic, the controller creates a new flow table. According to the linear regression model, the routing information between these switches changes. The suggested method aims to make data transmission from the host to the target computer as reliable as possible.

This ensures that path assignment decisions have already been made when traffic enters the network domain. Furthermore, the calculation workload on the network's routers will be reduced because of this. The suggested method uses source-to-destination pairings to estimate the delay for all possible paths in the network. As a result, the linear regression model must learn both incoming traffic characteristics and current network circumstances. Source and destination Internet Protocol (IP) addresses, size of the packet, number of packets, requested load, data rates, and available routes from the source to destination pair are inputs from the entrance port router.

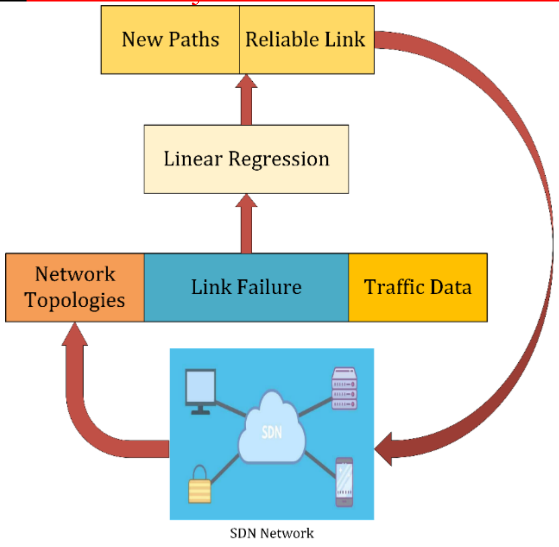

Figure 3 Proposed Methodology Diagram

Furthermore, two critical network domain inputs are considered: congestion rate and available routes, which can be classified as open or broken links. Based on all the information acquired by the suggested solution, the algorithm's output will be the predicted delays for all available routes. Incoming traffic will subsequently be assigned to the route with the smallest estimated delay by the route computation module. Because the suggested technique will compute the shortest path, all traffic delays are expected to be reduced. Because delay is inversely related to throughput, throughput is expected to improve.

A flowchart depicting how the entire network works are shown in Figure 4. Suppose paths between the host and destination computers are congested in this state diagram packet delivered with the controller's algorithm. In that case, the controller switches to the proposed algorithm and uses this method to find a new alternating path. The packets are sent properly if the controller identifies network links that are not congested.

Figure 5 explains our flow of study. First, we collect datasets from an internet resource. After pre-processing the dataset, we extract features from the dataset and then we select features for our research work. We train and test our dataset using a Machine Learning algorithm. This model predicts values and helps us for finding alternative routes.

Simulation System.

To analyze the justification of the intended algorithm, we must first define the simulation environments and parameters listed in Table #1. This table covers the virtual network as well as all the simulation's parameter values. We utilized the Mininet tool to simulate the situation[29]. This virtual network has one SDN controller, six switches, five computer systems, and about 23 edges that connect the switches and computer systems to the controller.

In the simulation, we use the network structure shown in Fig 2. We must decide to take the bandwidth of every edge 800Mbit to transmit data from one point to another under the same situation.

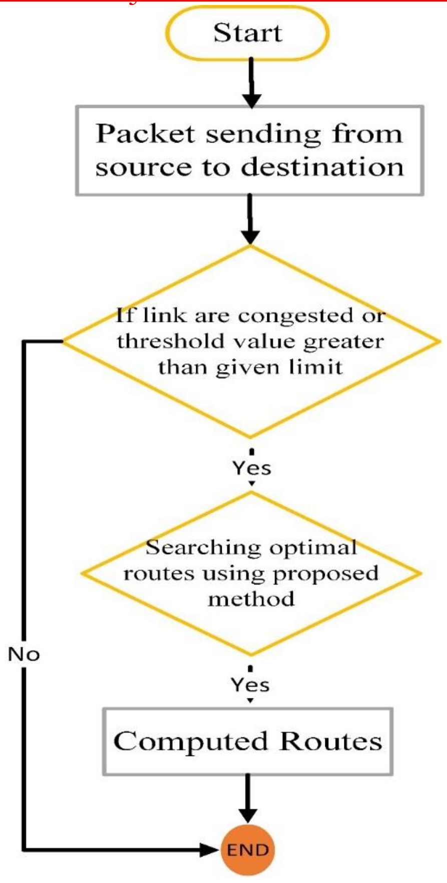
Start
Packet sending from source to destination
If link are congested or threshold value greater than given limit
Yes
Searching optimal routes using proposed method
Yes
Computed Routes
No
END

Figure 4 Flowchart of Proposed Method

The Dataset.

We collect our SDN dataset for simulation purposes from a very well-known website. We check and compare our dataset with many other datasets and find our dataset very best for our simulation. This dataset contains 134000 rows and 16 columns. We save this dataset in a CSV file and extract the necessary features that can be used in our simulation[30]. We delete unnecessary features from the dataset. We choose the columns to identify the different features, such as switch_id, total_time, packet_in, packet_loss, etc, as input and labels for output. We choose 70% data for training purposes and based on this training. We choose 30 % data for testing purposes.

Results and Discussion.

The experimental findings exist in this section. The tests are run on an Intel Core i5 CPU running Python 3.7. The python code was executed, and results were obtained using the Jupyter notebook and Mininet emulator[31]. Our method was implemented in a Pox controller responsible for network monitoring, backup path computation, and forwarding rules in switches[32]. The controller was installed in its virtual machine and is linked to another virtual machine that runs the Mininet emulation platform. To generate traffic between the nodes in the networks, the iperf3 program was utilized[33]. Wireshark collected traffic statistics in normal and link failure scenarios, including aggregate flow size, round-trip duration, and packet loss ratio. On the Mininet platform, we randomly disconnected a link in the network topology to mimic a link failure while traffic flows were redirected across the network. The controller's pre-installed backup path will be used to deflect all affected flows.

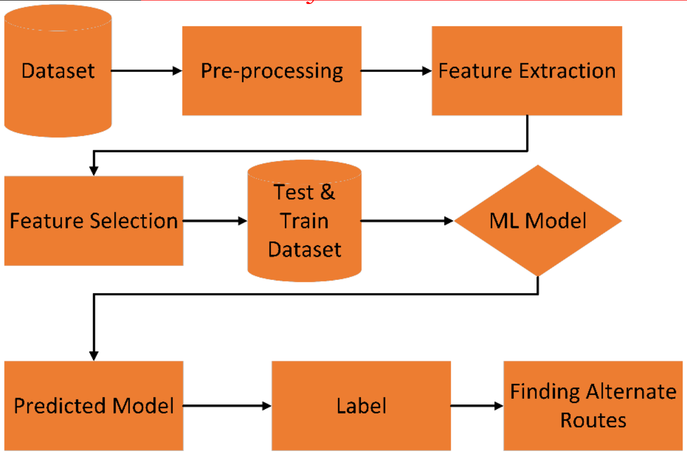

Figure 5 Flow of Study Diagram

Table 1.

Constraints	Values
SDN Controller	1
Nodes	5
Switches	6
Edges	23
Simulation Tool	Mininet
Bandwidth on edges	800Mbit
File Size	1GB to 3 GB

The recovery time and packet drop ratio in the event of a connection breakdown are used to gauge how well the proposed system performs. To demonstrate the level of reduction in recovery time, the suggested system is compared to the existing Segmentation technique. The time how long it takes for packet transmission to resume after a network link failure is recorded.

In figure 6, there is a comparison of flow entries between the proposed solution and both FTLink and Shortest Path Rotes Algorithm (SPRA) techniques. Since there are fewer entries in the flow table, there is less memory overhead associated with the controller's full installation of flow entries in the switches[34].

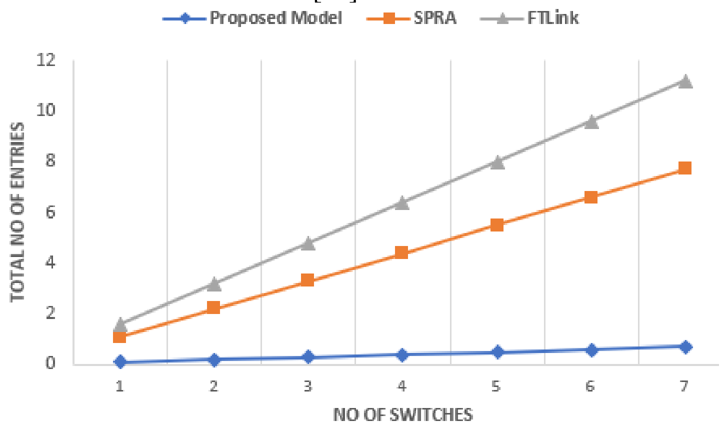

Figure 6 Comparison of flow entries b/w proposed method with FTLink and SPRA

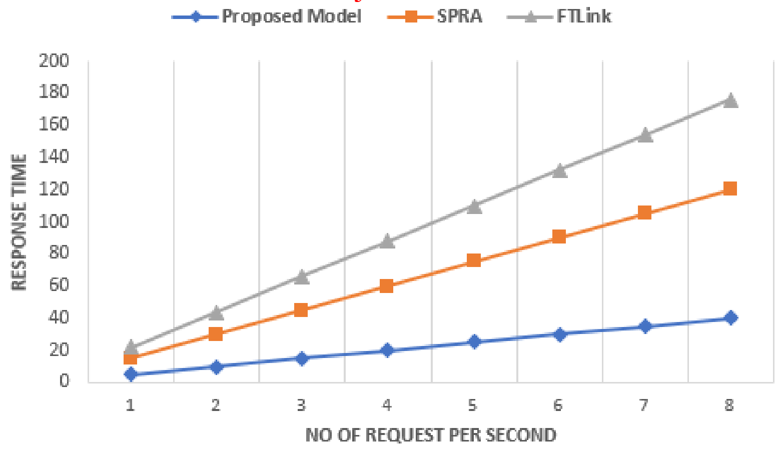

Figure 7 Comparison b/w performance of the proposed method with FTLink and SPRA

Figure 7 shows a comparison of the proposed approach's performance with that of the FT Link and SPRA methods[35]. It can be shown that the proposed method performs better than either technique.

Here we must compare latency between the proposed solution and both state-of-the-art methods. Because traffic is routed across numerous channels rather than depending on a single lane after failure, as seen in Fig. 8, the SPRA and FTLink have higher latency after failure, whereas the proposed solution's latency shows identical statistical performance before and after failure, as shown in Fig. 9.

Table # 2,3,4, and 5 describes all the simulation and comparison results among FTLink, SPRA, and proposed techniques.

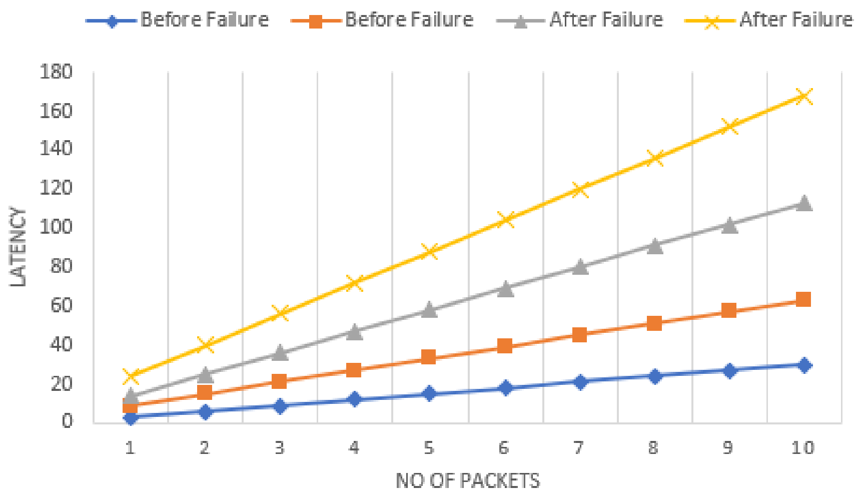

Figure 8 Latency with FTLink and SPRA

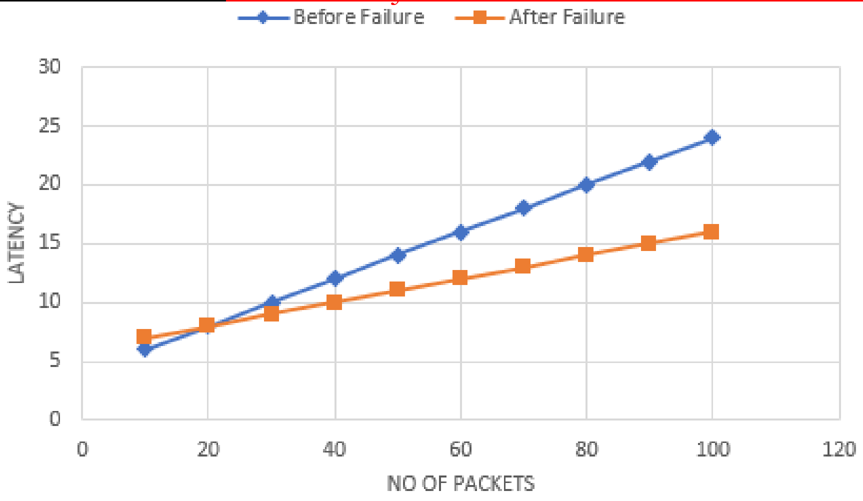

Figure 9 Latency with a proposed method

Table 2. This table describes the flow entries between the state-of-the-art and proposed model.

Flow Entries					
SPRA		FTLink		Proposed	
No of Switches	Total no of Entries	No of Switches	Total no of Entries	No of Switches	Total no of Entries
7	7	7	3.5	7	0.7

Table 3. This table describes the performance of the network between the state-of-the-art and proposed model.

Performance					
SPRA		FTLink		Proposed	
Request made per second	Response Time	Request made per second	Response Time	Request made per second	Response Time
8	80	8	56	8	40

Table 4. This table describes the latency of the network with FTlink and SPRA before and after failure.

Latency with FTLink and SPRA			
Before Failure		After Failure	
No of Packets	Latency	No of Packets	Latency
100	30	100	50

Table 5. This table describes the latency of the network with a proposed model before and after failure.

Latency with Proposed Method			
Before Failure		After Failure	
No of Packets	Latency	No of Packets	Latency
100	32	100	42

Conclusion.

Network link failure is a major problem that must be identified and addressed soon to prevent from loss of data packets. In the network connection, failure is caused by traffic burden and a switch topology loop. Connection failures harm network dependability. Recovery time must be kept to a minimum for the network to be more dependable and efficient. The proposed technique manages packets with the least amount of latency when a network link fails by calculating a list of shortest routes and moving packets from the failure location. The controller aids in the computation of the shortest paths quickly, which may aid in the recovery of problems in the computed backup paths. Once a network has recovered from a loss, the proposed technique can handle failures in additional network connections. The existing SDN controller technique increases the number of flow entries by adding both primary and backup path entries to the switch. The suggested technique installs flow entries only when they are needed, resulting in fewer flow entries and decreased memory usage. According to the simulation results, the proposed method improves performance by decreasing the number of flow entries in the switches during flow computation and speeding up recovery.

More bandwidths will be used in the future, which may be accomplished by transmitting packets with the best bandwidth while considering the combined capacity of all links. The technique can be used in wireless mobile networks to provide authorized link connectivity between mobile nodes while considering the packet loss ratio. In the event of a link loss, the suggested approach enables a quick switchover between base stations.

REFERENCES

[1] M. Ahmid, O. Kazar, and L. Kahloul, "A secure and intelligent real-time health monitoring system for remote cardiac patients," *Int. J. Med. Eng. Inform.*, vol. 14, no. 2, pp. 134–150, 2022, doi: 10.1504/IJMEI.2022.121130.

[2] Y. Wang, D. Jiang, L. Huo, and Y. Zhao, "A New Traffic Prediction Algorithm to Software Defined Networking," *Mob. Networks Appl. 2019 262*, vol. 26, no. 2, pp. 716–725, Dec. 2019, doi: 10.1007/S11036-019-01423-3.

[3] A. N. Shahbaz, H. Barati, and A. Barati, "Multipath routing through the firefly algorithm and fuzzy logic in wireless sensor networks," *Peer-to-Peer Netw. Appl. 2020 142*, vol. 14, no. 2, pp. 541–558, Oct. 2020, doi: 10.1007/S12083-020-01004-2.

[4] S. H. Haji *et al.*, "Comparison of Software Defined Networking with Traditional Networking," *Asian J. Res. Comput. Sci.*, no. June, pp. 1–18, 2021, doi: 10.9734/ajrcos/2021/v9i230216.

[5] D. S. Rana, S. A. Dhondiyal, and S. K. Chamoli, "Software Defined Networking (SDN) Challenges, issues and Solution," *Int. J. Comput. Sci. Eng.*, vol. 7, no. 1, pp. 884–889, 2019, doi: 10.26438/ijcse/v7i1.884889.

[6] A. Shaghaghi, M. A. Kaafar, R. Buyya, and S. Jha, "Software-Defined Network (SDN) data plane security: Issues, solutions, and future directions," *Handb. Comput. Networks Cyber Secur. Princ. Paradig.*, pp. 341–387, Jan. 2019, doi: 10.1007/978-3-030-22277-2_14/COVER.

[7] J. Ali, G. M. Lee, B. H. Roh, D. K. Ryu, and G. Park, "Software-defined networking approaches for link failure recovery: A survey," *Sustain.*, vol. 12, no. 10, 2020, doi: 10.3390/su12104255.

[8] H. A. Eissa, K. A. Bozed, and H. Younis, "Software Defined Networking," *19th Int. Conf. Sci. Tech. Autom. Control Comput. Eng. STA 2019*, no. March 2019, pp. 620–625, 2019, doi: 10.1109/STA.2019.8717234.

[9] K. Bakhshi Kiadehi, A. M. Rahmani, and A. Sabbagh Molahosseini, "A fault-tolerant architecture for internet-of-things based on software-defined networks," *Telecommun. Syst.*, vol. 77, no. 1, pp. 155–169, May 2021, doi: 10.1007/S11235-020-00750-1.

[10] X. Gao, M. Qiu, and M. Liu, "Machine Learning Based Network Censorship," *2021 8th IEEE Int. Conf. Cyber Secur. Cloud Comput. (CSCloud)/2021 7th IEEE Int. Conf. Edge Comput. Scalable Cloud*, pp. 149–154, Jun. 2021, doi: 10.1109/CSCLOUD-EDGECOM52276.2021.00036.

[11] C. Janiesch, P. Zschech, and K. Heinrich, "Machine learning and deep learning," *Electron. Mark.*, vol. 31, no. 3, pp. 685–695, Apr. 2021, doi: 10.1007/s12525-021-00475-2.

[12] Q. M. A. and M. . Kiran. I, Siddique. Z, Butt. A. R, Mudassir. A. I, "Towards Skin Cancer Classification Using Machine Learning and Deep Learning Algorithms: A Comparison," *I nternational J. Innov. Sci. Technol.*, vol. 3, no. special issue, pp. 110–118, 2021.

[13] M. Alizamir, S. Kim, O. Kisi, and M. Zounemat-Kermani, "A comparative study of several machine learning based non-linear regression methods in estimating solar radiation: Case studies of the USA and Turkey regions," *Energy*, vol. 197, Apr. 2020, doi: 10.1016/J.ENERGY.2020.117239.

[14] S. Ray, "A Quick Review of Machine Learning Algorithms," *Proc. Int. Conf. Mach. Learn. Big Data, Cloud Parallel Comput. Trends, Prespectives Prospect. Com. 2019*, pp. 35–39, Feb. 2019, doi: 10.1109/COMITCON.2019.8862451.

[15] M. Anjum.S. M, Riaz.O,Latif, S, "Diastolic Dysfunction Prediction with Symptoms Using Machine Learning Approach," *Int. J. Innov. Sci. Technol.*, vol. 4, no. 3, pp. 714–727, 2022, [Online]. Available: https://journal.50sea.com/index.php/IJIST/article/view/280/

[16] A. Malik, B. Aziz, M. Adda, and C. H. Ke, "Smart Routing: Towards Proactive Fault-Handling in Software-Defined Networks," *Comput. Networks*, vol. 170, Apr. 2019, doi: 10.48550/arxiv.1904.00717.

[17] R. O. and S. W. Anjum. M. S, Mumtaz. S, "Heart Attack Risk Prediction with Duke Treadmill Score with Symptoms using Data Mining," *I nternational J. Innov. Sci. Technol.*, vol. 3, no. 4, pp. 174–185, 2021.

[18] L. El-Garoui, S. Pierre, and S. Chamberland, "A new sdn-based routing protocol for improving delay in smart city environments," *Smart Cities*, vol. 3, no. 3, pp. 1004–1021, 2020, doi: 10.3390/smartcities3030050.

[19] R. A. U. Ullah. A, Qayyum. H, Hassan. F, Khan. M. k, "Comparison of Machine Learning Algorithms for Sepsis Detection," *Int. J. Innov. Sci. Technol.*, vol. 4, no. 1, pp. 175–188, 2022, [Online]. Available: https://journal.50sea.com/index.php/IJIST/article/view/190

[20] S. Moazzeni, M. R. Khayyambashi, and N. Movahhedinia, "Improving the Reliability of Software-Defined Networks with Distributed Controllers Through Leader Election Algorithm and Colored Petri-Net," *Wirel. Pers. Commun.*, vol. 109, no. 1, pp. 645–656, Nov. 2019, doi: 10.1007/S11277-019-06583-9.

[21] R. A. Manzoor. S, Qayyum. H, Hassan. F, Ullah. A, Nawaz. A, "Melanoma Detection Using a Deep Learning Approach," *Int. J. Innov. Sci. Technol.*, vol. 4, no. 1, pp. 222–232, 2022.

[22] T. Hu, P. Yi, J. Lan, Y. Hu, and P. Sun, "FTLink: Efficient and flexible link fault tolerance scheme for data plane in Software-Defined Networking," *Futur. Gener. Comput. Syst.*, vol. 111, pp. 381–400, Oct. 2020, doi:

10.1016/J.FUTURE.2019.11.015.

[23] T. Truong-Huu, P. Prathap, P. M. Mohan, and M. Gurusamy, "Fast and adaptive failure recovery using machine learning in software defined networks," *2019 IEEE Int. Conf. Commun. Work. ICC Work. 2019 - Proc.*, May 2019, doi: 10.1109/ICCW.2019.8757169.

[24] M. Silva Freitas, R. Oliveira, D. Molinos, J. Melo, P. Frosi Rosa, and F. De Oliveira Silva, "ConForm: In-band Control Plane Formation Protocol to SDN-Based Networks," *Int. Conf. Inf. Netw.*, vol. 2020-January, pp. 574–579, Jan. 2020, doi: 10.1109/ICOIN48656.2020.9016580.

[25] R. B. Shohani and S. A. Mostafavi, "Introducing a New Linear Regression Based Method for Early DDoS Attack Detection in SDN," *2020 6th Int. Conf. Web Res. ICWR 2020*, pp. 126–132, Apr. 2020, doi: 10.1109/ICWR49608.2020.9122310.

[26] P. Kamboj and S. Pal, "Software-Defined Networking in Data Centers," *Softw. Defin. Internet Everything Springer*, pp. 177–203, 2022.

[27] A. Wang, Z. Zha, Y. Guo, and S. Chen, "Software-Defined Networking Enhanced Edge Computing: A Network-Centric Survey," *Proc. IEEE*, vol. 107, no. 8, pp. 1500–1519, Aug. 2019, doi: 10.1109/JPROC.2019.2924377.

[28] M. P. Nowak and P. Pecka, "Routing algorithms simulation for self-aware sdn," *Electron.*, vol. 11, no. 1, 2022, doi: 10.3390/electronics11010104.

[29] L.-D. Chou, Y.-T. Yang, Y.-M. Hong, J.-K. Hu, and B. Jean, "A Genetic-Based Load Balancing Algorithm in OpenFlow Network," pp. 411–417, 2014, doi: 10.1007/978-94-007-7262-5_48.

[30] Y. Huang and Y. Sun, "A Dataset of Daily Interactive Manipulation," *Int. J. Rob. Res.*, vol. 38, no. 8, pp. 879–886, Jul. 2018, doi: 10.48550/arxiv.1807.00858.

[31] M. Hasan, H. Dahshan, E. Abdelwanees, and A. Elmoghazy, "SDN Mininet Emulator Benchmarking and Result Analysis," *2nd Nov. Intell. Lead. Emerg. Sci. Conf. NILES 2020*, pp. 355–360, Oct. 2020, doi: 10.1109/NILES50944.2020.9257913.

[32] M. Y. Daha, M. S. M. Zahid, B. Isyaku, and A. A. Alashhab, "CDRA: A Community Detection based Routing Algorithm for Link Failure Recovery in Software Defined Networks," *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 11, pp. 712–722, 2021, doi: 10.14569/IJACSA.2021.0121181.

[33] I. Khan and K. Chen, "EBA: Efficient Bandwidth Aggregation for Connected Vehicles with MPTCP," *IEEE Internet Things J.*, vol. 9, no. 8, pp. 5812–5823, Apr. 2022, doi: 10.1109/JIOT.2021.3065911.

[34] S. Petale and J. Thangaraj, "Link Failure Recovery Mechanism in Software Defined Networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 7, pp. 1285–1292, 2020, doi: 10.1109/JSAC.2020.2986668.

[35] R. H. Jhaveri, R. Tan, A. Easwaran, and S. V. Ramani, "Managing industrial communication delays with software-defined networking," *Proc. - 2019 IEEE 25th Int. Conf. Embed. Real-Time Comput. Syst. Appl. RTCSA 2019*, Aug. 2019, doi: 10.1109/RTCSA.2019.8864557.