Layer 2 Path Evaluation System Using Machine Learning

Mahamah Sebakor, Non-member

ABSTRACT

Does it seem strange that the spanning tree protocol (STP) has so far been the only layer-2 backbone defense against looping? Do we trust it? For several decades, the campus backbone has often been an unsuspected problem in STP failure. Meanwhile, MAC address flapping is probably a feasible way of solving the issue for modern network fabrics. Due to the serious issues involved with layer-2, particularly the legacy switches in the extended STP design, this work uses the notion of a software-defined network to evaluate both traditional and modern networks. Through the MAC address lookup for all bridge devices, this work proposes a Layer-2 evaluation system (LES), which uses a novel approach known as support supervised learning to create the data preparation for machine learning. Additionally, the LES enabled network allows administrators to determine their backbones. This study aims to evaluate the potential slowdown in the network caused by MAC address problems. Furthermore, it investigates the proposed method in a real network, while also covering the evaluation and performance of our proposed method.

Keywords: Spanning Tree Protocol Failure, STP Failure, Network Evaluation, Ethernet, Bridging Loop, Machine Learning

1. INTRODUCTION

The spanning tree protocol (STP) in the Ethernet network has been in widespread use for some time. Currently, the new modern network is a hot research topic in the field of network technology, including the Ethernet network fabric, software-defined network (SDN), and network function virtualization (NFV). However, the use of traditional networks remains popular. Previously, the STP was essential in preventing the loop in the layer-2 network once redundant links were implemented. In this section, we briefly discuss related technologies, including the spanning tree algorithm (STA), SDN, graph, and machine learning.

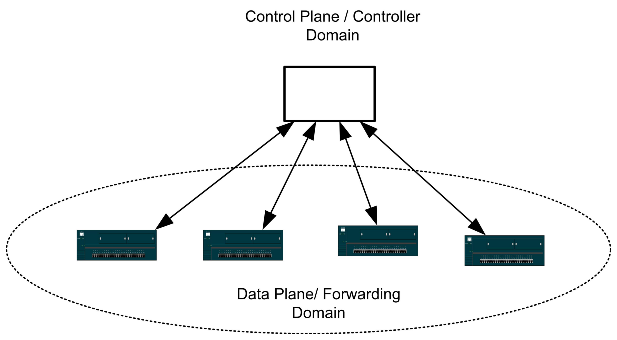
Control Plane / Controller Domain
Data Plane/ Forwarding Domain

Fig. 1: *Software-defined network concept.*

The STP initiates a topology path by selecting the root switch, root port, and designated port. Occasionally, the failure of the algorithm itself probably imposes disturbances and irregularities in the campus network. Technically, the STP uses bridge protocol data units (BPDUs), distributed from the root down to the edge switches. Thereafter, each switch uses a flood and learns the MAC address. Since the layer 2 Ethernet does not provide time-to-live (TTL) to protect the switching loop-like layer 3 technology, the layer 2 network might lead to tremendous traffic in the network or even layer 2 traffic looping once STP failure occurs. According to previous studies, one of the common reasons for STA failure is the uncontrollable loss of BPDUs, leading to a blocked port transforming to the forwarding mode. This occurs when the switch event is caused by a duplex mismatch, unidirectional link, packet corruption, resource errors, port fast configuration error, diameter issues, or software errors. Another typical issue is MAC address flapping or an unstable MAC address table, potentially caused by wireless mobility, fake MAC addresses, and MAC address spoofing. In the case of an administration configuration to their local area network (LAN) infrastructure, such as a disabled STP, uncountable traffic will occur in the network, eventually becoming a loop.

The SDN is a controller-based concept. As usual, the data traffic is never passed to the controller. The forwarding domain is the real traffic conveyance, consisting of two main planes: control and data, as presented in Fig. 1. Briefly, a legacy device has its

Manuscript received on January 25, 2021 ; revised on July 13, 2021 ; accepted on August 19, 2021. This paper was recommended by Associate Editor Nattapong Kitsuwan.

The author is with the School of Information Technology, Mae Fah Luang University, Chiang Rai, Thailand.
Corresponding author: mahamah.seb@mfu.ac.th

Digital Object Identifier 10.37936/ecti-eec.2021193.244943

own small brain inside, known as a control plane, and also forwarding capability. A novel approach to a traditional network known as an SDN moves the small brain to the outside, in a similar way to a birds-eye view. In addition, the programmability of traffic engineering, security, and monitoring are also applicable to SDN functions.

Discrete math, particularly graph theory, was first discussed in 1736. A graph is a collection of edge sets (E), vertices (V), and faces (F). Edges are well-known links or connections, whereas vertices are nodes in the network field. The combination of sets E and V is known as a graph (G); thus, we can say that $G = E + V$. However, the faces are similar to those in the regions. According to Euler's formula, the number of V minus E plus F is always equal to two, such that $V - E + F = 2$.

Thus, a graph constitutes a network, divided into two major types: (1) direction and (2) undirection. The cardinality of the graph $|G|$ is the number of vertices, whereas the degree is the number of edges initiated from a vertex. Therefore, it can be said that $\deg(V_i)$ is the number of degrees in each vertex. Moreover, the nodes that sit nearby are known as adjacency matrices, and an isomorphic graph could be said to resemble the idea of a tree, known as a spanning tree. In general, any graph can be recognized as cycles, spanning trees, and dual graphs. The spanning tree graph can identify the number of vertices using the following formula: $E + 1 = V$.

Several machine learning algorithms exist, such as decision trees, support vector machines, and random forests. Machine learning is the most popular tool used for system analysis. It provides various categories of learning models, including supervised and unsupervised learning. The supervised learning model has an outstanding classification and regression function. In particular, the decision tree comprises the root, branches, decision nodes, and leaves, which are the main components of these algorithms. The results of the prediction appear on the leaf. Iterations ensured that the prediction will be answered correctly. The decision tree starts with a splitting method by selecting the highest information gain (low entropy/Gini) to determine the split decision node. The split is then repeated using identical criteria until the end. In this study, a decision tree is used to solve classification problems, providing a supervised learning technique for prediction. Use of the proper training dataset results in the best model, which is the main aim output of these learning steps. Furthermore, this machine learning algorithm can be applied to predict unknown factors.

To satisfy these requirements, this study proposes network automation software known as a layer-2 evaluation system (LES). The LES retrieves information from the real contiguous layer-2 switches of the network. Subsequently, semi-supervised learning (SSL) generates a training sample set, evaluates its accuracy, and then predicts the issue. However, this SSL is inspired by graph theory. The method proposed in this study is also applied to the Mea Fah Luang University network (MFU) for monitoring, evaluating, and improving network performance. Moreover, this study can also be applied to both basic structures and advance networks. Briefly, the proposed method is composed of (1) a MAC address lookup and (2) an analytical system.

The remainder of this paper is organized as follows: Section 2 presents a comprehensive summary of the related works, while the proposed approach is described in Section 3. The statements of performance and evaluation are then revealed in Section 4, while Section 5 presents the discussion and conclusion, and suggestions for future work.

2. RELATED WORK

Various technologies have been developed in relation to this work such as STA failure, MAC address flapping, bridging loops, graph theory, fabric networks, SDNs, and machine learning.

Several approaches have been proposed to evaluate and analyze regular issues in the network architecture. Some researchers are concerned about the broadcast storm of layer-2 looping [4] or even propose loop-free avoidance, which uses a meshed tree protocol (MTP) [1]. A few studies have developed graphs and solved the graph problem [20], including proposing an algorithm for a weighted undirected graph [16]. Another proposed method involves creating a reduced graph by employing vertex merging technology [19]. Meanwhile, in [7], the number of minimum spanning tree (MST) edges is reduced through the disjoint algorithm. This study also examines the new modern network fabric technology. For instance, the researchers in [12] proposed the use of fabric-based architecture in the carrier network, where the fabric is composed of leaf and spine, whereas [13] uses the SDN concept for a new network fabric. Furthermore, several researchers exploit the graph to the real world, such as in [2], using the MST of the graph in the transportation system; in [3], the authors considered a spanning tree of node failure by using the graph-based method, whereas in [18], the issue of expanding the spanning tree in graphs and solving the CST problem are addressed.

Recently, a new term SDN has been frequently expressed. In [11], a spanning tree is constructed as a controller and a DDOT created. In [15], a method for binding legacy spanning trees with SDN central control is proposed, while the researchers in [14] report on the migration between SDN and STP. In addition, the combination of SDN and machine learning has been discussed [5], along with applying

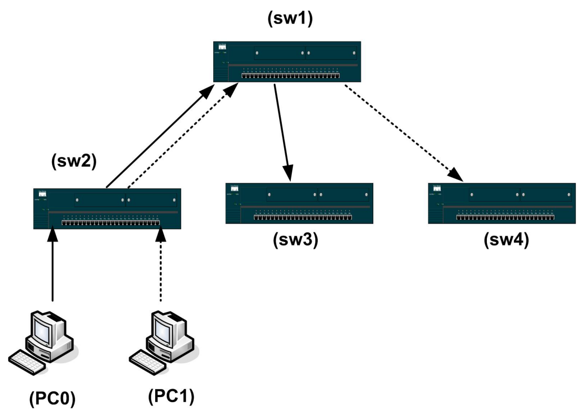

Fig. 2: *Flood and learn concept.*

it to the network. In [8], the authors briefly discuss various machine learning methods for application in the network field [9] using machine learning to detect the optical layer. Some previous studies are similar to this work, such as [17], which uses machine learning for dynamic routing in real time, whereas in [10], Syslog is used to classify the root cause of the network. However, these are layer 3 traffic optimizations and do not use machine learning for classification.

Machine learning and SDN both play a dominant role in modern networks. However, traditional layer-2 networks also remain popular. To address the solution and evaluation of common issues, several researchers have focused on STP rather than the entire network, applying machine learning to some events, and using graphs for others. In contrast, this work focuses on finding the untypical layer-2 network by looking at the output of STP and evaluating the STP results.

3. PROPOSED APPROACH

Of the methodologies in existence, the design of this work belongs to the SDN. The main objectives of this study are to classify and predict potential problems. For classification purposes, a supervised learning technique is created. The main idea is that at a certain time or within a small timeframe, only one unique MAC address can appear in the entire backbone. The LES, which is similar to graph theory, is composed of (1) SSL algorithm and (2) the analytical process. Briefly, the SSL algorithm uses the SDN concept to retrieve data from a real-world forwarding domain. It then prepares the data for training and application. Eventually, machine learning algorithms are used for analysis and evaluation.

Before considering the algorithm, the MAC address flood and learn concept should first be discussed. The term BUM may be familiar to some readers, it represents broadcast, unknown unicast, and multicast traffic. When broadcast traffic occurs, it is necessary to switch forward out to all ports within the same spanning tree instance except for the incoming port, as demonstrated in Fig. 2. Assuming that sw1 is the root of a spanning tree, and two instances are created, the first is the dark line, while the second is the dotted line. When sw2 receives the broadcast traffic from pc0, it will forward that traffic out to every port which is a member of the dark line except the incoming port; in this case, only sw1. Thereafter, sw1 forwards the traffic out to other member ports of the dark line, namely, sw3. Therefore, this is the first topology for the first instance. Similarly, once pc1 sends a broadcast to sw2, sw2 then forwards the traffic out by taking the second instance, that is, sw1, and thereafter sw1 to sw4.

3.1 SSL Algorithm

Taking a closer look at each part of the algorithm, rather than using BPDU communication at the local knowledge level, this study employs the central controller by looking up all MAC addresses in an entire network, which is slightly similar to the bird's eye view concept. It then creates training and testing data and subsequently predicts an unlabeled attribute. Two scenarios are considered: (1) a single switch and (2) a layer-2 virtual switch (VS).

3.1.1 Single Switch

In this scenario, the traffic direction is divided into two major switches: root and leaves, which are similar to the tree-based concept. Unlike the STA root switch, the root switch in this study may offer multiple port designs along the bottom of the branches, where the remaining branch switches have only one root port point to the root switch. Furthermore, STP topology has only one root switch in each STP instance. Once the STP forwards the traffic, the MAC addresses use flooding to create the MAC address table. Every switch has a unique port for each MAC address destination, whereas each STP instance has a unique path topology. In contrast, this study sets a root (R) as the node sitting nearest to the destination of the MAC address (m). Technically, at time t, each m will have its topology denoted by $TP(t)$, and TP comprises R and leaves (L). As presented in Fig. 3(a), four interfaces, namely $\{e1, e2, e3, e4\}$, are connected to a switch; m_1 locates at $e1$. Thus, TP_1 (dark line) is a topology of m_1 comprising the root R_1 and the leaves L_{11} and L_{12}, as indicated in Fig. 3(b). Similarly, TP_2 (the dotted line) is an m_2 topology. Notably, it is impossible for a switch to forward traffic via multiple interfaces to the same MAC address destination simultaneously. For instance, in the case of $m_1 = m_2$, it implies that there are two TPs and two Rs at time (t). In a similar way, in a smaller timeframe $TP\ (\Delta t)$, if the interface of m_1 moves from $e1$ to $e2$, then swaps back to $e1$; in this

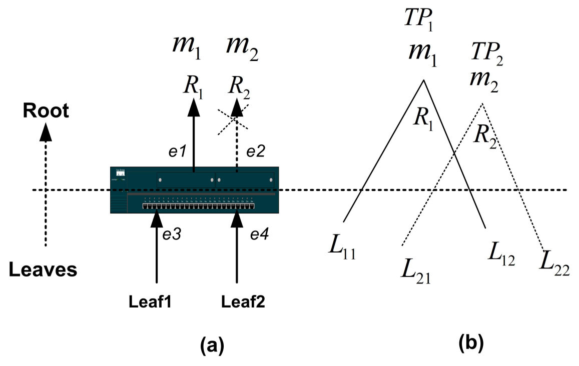

Fig. 3: *Single switch and its MAC topology.*

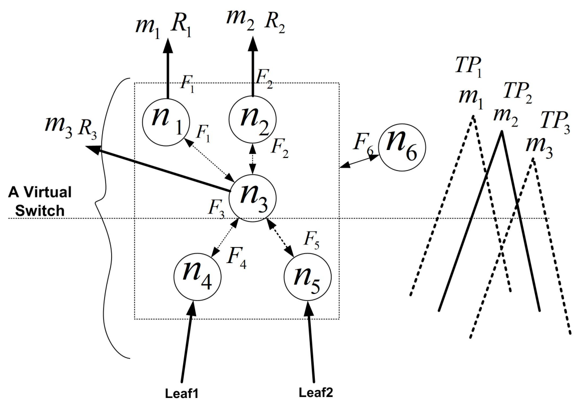

Fig. 4: *Virtual switch and its MAC topology.*

case, the number of topologies, $K(TP)$, as shown in Eq. (1), is two, where the cardinality of TP represents a topology.

$$K(TP) = |TP_i| \,. \tag{1}$$

Furthermore, MAC address flapping or MAC address table instability is a common problem. Hence, if the number of topologies is greater than 1, $K(TP) > 1$, it will be recognized as abnormal.

3.1.2 Virtual Switch

Analogous to a single switch, in this scenario, all devices located inside the domain of interest are treated as a single switch, as illustrated in Fig. 4. Five nodes are located in a rectangle, with one node isolated. The rectangle is most likely to be the domain of interest (ID) or VS. In similarity to a single switch, each root-leaf-based topology of each MAC address will have only one root, or it can be said that $K(TP) = 1$. Fig. 4 shows that three root-leaf-based topologies occur. Each TP, host m_1, m_2, and m_3 has its nearest root as R_1, R_2, and R_3, respectively. In some circumstances, it can be assumed that for a Δt range in which $\Delta t = t_1 - t_0$, if $m_3 = m_2 = m_1$, then $K(TP) = 3$. In this case, it implies that the m-MAC address is unusual or flapping.

The network model (NM) is a collection of a pair of nodes and forwarding interfaces NM $= \langle N, F\rangle$, where

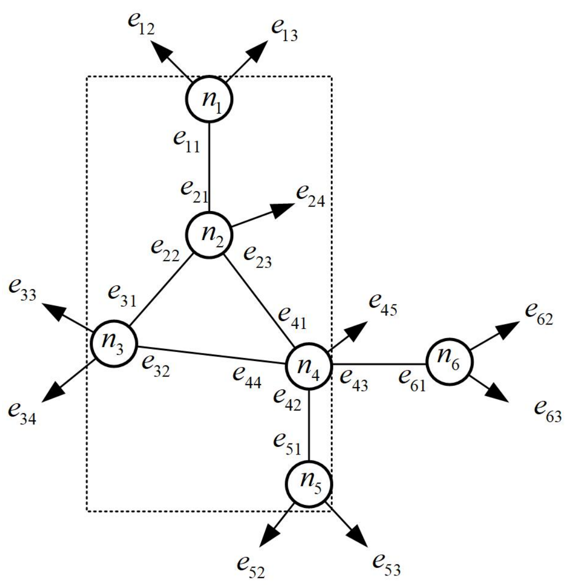

Fig. 5: *Simulation and in-band links.*

N denotes the set of nodes of the entire network, and F the set of forwarding interfaces or links of the NM. In this work, the nodes are categorized into (1) in-band nodes (iN) and (2) out-of-band nodes (oN). The term iN denotes the set containing elements located inside the ID, whereas oN is located outside the domain of interest. Therefore, $N = iN \cup oN$ and $iN = \{n : n \in N, n \notin oN\}$. It can therefore be said that iN is the set containing elements of nodes such that nodes are devices located in the network, under the condition that they are not elements or located outside the domain of interest. Thus, a VS is a combination or collection of connected nodes and forwarding interfaces that visually sit inside the domain ID = $\langle iN, iF\rangle$ of interest.

Similarly, $F = \{F_1, F_2, \ldots, F_n\}$ where all elements are sets of forwarding interfaces of each node n. Meanwhile, the forwarding interfaces of each node are $F_n = \{e_{n1}, e_{n2}, \ldots, e_{nj}\}$, where n represents the node number identification $\forall F_n \in F$. Similarly, two categories of F are considered. The forwarding interfaces connecting node to node inside the ID are known as in-band-forwarding interfaces (iF), while the remaining interfaces are out-of-band-forwarding interfaces (oF). The term iF denotes the set containing the elements of the interfaces. These elements are sets containing the connected interfaces in which each element is a subset of F_n and under the condition that the sets are elements of VS or $iF = \{E : E_n \subseteq F_n, \forall E_n \in VS\}$. However, VS will be discussed later.

The algorithm will now be considered in detail using an example for clarity. In Fig. 5, the NM comprises six nodes $N = \{n_1, n_2, \ldots, n_6\}$, five of which are in-band nodes $iN = \{n_1, n_2, \ldots, n_5\}$, while one is an out-of-band node $oN = \{n_6\}$.

Table 1: *Example set at* Δt_1.

VL	MAC	n_1	n_2	n_3	n_4	n_5	Status
VL_1	m_{11}	e_{11}	e_{24}	e_{31}	e_{41}	e_{51}	0
VL_1	m_{12}	e_{11}	e_{22}	e_{35}	e_{43}	e_{51}	1
VL_2	m_{21}		e_{23}		e_{43}		0
VL_2	m_{21}		e_{24}	e_{33}			1

The forwarding interface of an individual node is $F = \{\{e_{11}, e_{12}, e_{13}\}, \{e_{21}, e_{22}, e_{23}, e_{24}\}, \ldots, \{e_{61}, e_{62}, e_{63}\}\}$, along with the in-band interface $iF = \{\{e_{11}\}, \{e_{21}, e_{22}, e_{23}\}, \{e_{31}, e_{32}\}, \ldots, \{e_{41}, e_{42}, e_{43}, e_{44}\}, \{e_{51}\}\}$, where e denotes the set containing the elements of adjacency nodes $e_{11} = \{n_1, n_2\}$, $e_{21} = \{n_2, n_1\}$, $e_{22} = \{n_2, n_3\}$, $e_{23} = \{n_2, n_4\}$, and $e_{31} = \{n_3, n_2\}$. Furthermore, let VL_i and m_i be the VLAN and MAC addresses, respectively. The main core algorithm of this work is divided into two tasks: (1) the random creation of a dataset and (2) label identification.

Prior to undertaking these tasks, the LES gathers all necessary information from the ID, such as MAC address m_i, VLAN number VL_i, nodes iN, and interfaces F_n. At that point, the LES generates the iF set. Next, a VS metric is created by setting the ID into a metric. A matrix is subject to both the network diagram and iF set, as in Eq. (2).

$$VS = \begin{bmatrix} 0 & e_{11} & 0 & 0 & 0 \\ e_{21} & 0 & e_{22} & e_{23} & 0 \\ 0 & e_{31} & 0 & e_{32} & 0 \\ 0 & e_{41} & e_{44} & 0 & e_{42} \\ 0 & 0 & 0 & e_{51} & 0 \end{bmatrix}. \tag{2}$$

The VS metric illustrates a 5×5 metric, which is composed of the adjacency interfaces. Rows and columns refer to the directions of the adjacency node; for instance, at the point where the row is 1 and column is 2, is represented as node n_1 connecting to n_2 such that the adjacency interface is e_{11} or it could be said that $VS_{1,2} = e_{11}$. Similarly, the remaining matrices are $VS_{2,1} = e_{21}$, $VS_{2,3} = e_{22}$, $VS_{2,4} = e_{23}$, $VS_{3,2} = e_{31}$, $VS_{3,4} = e_{32}$, $VS_{4,2} = e_{41}$, $VS_{4,3} = e_{44}$, $VS_{4,5} = e_{42}$, and $VS_{5,4} = e_{51}$. It should be noted that n_6 and e_{43} are not located inside the VS matrix.

Taking a closer look at the first task, this algorithm starts by selecting the first MAC address and then randomly assigns a possible interface to each iN from the set F_n, as summarized in Table 1 with each row representing an example. For instance, in the first example, VL_1 and m_{11} are selected. The algorithm then randomly walks through each iN node by starting at the n_1 node. It then assigns n_1 an interface selected from F_1, that is e_{11}, e_{12}, e_{13}. Next, it walks through n_2 and performs the same task repeatedly until reaching n_5. Eventually, the random process is revealed, as presented in the first row of Table 1. The algorithm consecutively repeats a similar task on the next data example until it reaches the latest m_{in}. Under uncertain conditions, the non-interface is probably selected, as can be observed in the third row.

$$\underset{\text{(a)}}{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}} \qquad \underset{\text{(b)}}{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}}$$

$$\underset{\text{(c)}}{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}} \qquad \underset{\text{(d)}}{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}}$$

Fig. 6: *(a) R and (b) TP metric for Table 1, row 1, (c) R and (d) TP metric for Table 1, row 2.*

Let us analyze the label identification procedure. In the first task, the algorithm initially finds the root R and creates the topologies for each example set. If the random interfaces of n_i are not the elements of VS, then n_i will be marked as R, and this algorithm constructs the R and TP metrics.

At the top of Table 1, the first example set states that in Table 1, row 1, the n_2 interface is illustrated as e_{24}, which is not an element of VS. Therefore, n_2 will be the root or R. The R metric is shown in Fig. 6(a), while the TP metric is shown in Fig. 6(b). According to Fig. 6(a), the row represents the R location, while the columns represent the node members of this topology. Thus, the members of the topology are $\{n_1, n_2, n_3, n_4, n_5\}$. Fig. 6(b) shows the TP metric. According to Table 1, row 1, column 3 is e_{11} where $e_{11} = VS_{1,2}$ or it can be said that $e_{11} = \{n_1, n_2\}$. Given that column 4 is e_{24}, this point is the root of the topology. Then there is e_{31} where $e_{31} = VS_{3,2}$ or $e_{31} = \{n_3, n_2\}$. Next is e_{41} where $e_{41} = VS_{4,2}$ or $e_{41} = \{n_4, n_2\}$. The last is e_{51} where $e_{51} = VS_{5,4}$ or $e_{51} = \{n_5, n_4\}$. Hence, the elements of the metric TP_1 are $TP_1(1,2)$, $TP_1(3,2)$, $TP_1(4,2)$, and $TP_1(5,4)$. It should be noted that these are the topologies of the MAC address. The number of roots in each example is determined using Eq. (3).

$$K_n(R) = \sum |R_i|. \tag{3}$$

Let $|R_i|$ be the number of topology roots I, for instance n. If $K_n(R) = 1$, $K_n(TP) = 1$. It can be recognized that this example is a normal circumstance, and the remaining criteria can be considered abnormal. For instance, when $K_n(R) = 0$, a loop may occur. In this case, $K_n(R) > 1$ will implicitly imply flapping, a fake MAC address, or system error. Therefore, this algorithm adds either an irregular '1' or normal '0' label to the target factor for each individual example. For instance, in the first example, Table 1, row 1; $K_1(R) = 1$ and $K_1(TP) = 1$. Thus, this situation implies a suspicious activity. Multiple topologies may sometimes occur, as in the example provided in the second row. The R metric, as shown in Fig. 6(c), demonstrates that n_3 and n_4 are concurrent roots. The members of R_1 are $\{n_1, n_2, n_3\}$, while those of R_2 are $\{n_4, n_5\}$. Whereas, Fig. 6(d) indicates the TP metric of this example. According to the R metric, the first three rows are TP_1, while the remainder of the metric is TP_2. It should be noted that, $TP_1(3,:)$ is 0, which indicates that n_3 is a root such that $K_2(R) = 2$ and $K_2(TP) = 2$. Therefore, this example implicitly implies an abnormal situation. In Table 1, row 3, the blank columns do not represent missing data; nevertheless, this implies that there is nothing on the forwarding interface of this node. Therefore, these blanks do not handle missing data points in the decision tree. However, from this algorithm, it can be said that $K_3(R) = 1$ and $K_3(TP) = 1$. Finally, according to row 4 of the example, $K_4(R) = 2$ and $K_4(TP) = 2$, implying a network anomaly.

3.2 Analytical Process

Machine learning tools are used in this study, specifically, decision trees. The question is why the chosen algorithm is used; the answer will be provided later. Decision tree classification begins with the root node split/prune decision nodes and ends at the leaf nodes. At each leaf, the answer is given. This work imposes proper attributes or variables on the example set. The example sets are based on the researcher's previous methodological algorithm; however, this work retrieves the information from the real network to create the model. Thereafter, the example of the proposed model is split into 70:30% of the total for training and testing. Labeling the target variables of both the training and testing datasets is the first task, with the predict attribute used as a class label. Next, the model is constructed using a decision tree. The output model of the example set is compared with the original example set. In other words, the accuracy of each iteration is evaluated and validated. To achieve the best model, in similarity to the task performed here, the process is repeated until the best model is achieved. Consequently, the highest-quality model is eventually constructed. The missing attribute is predicted as I (1) or N (0). It can be observed from Table 1, that the status column or variable is a target binary dependent variable. While the rest are nominal independent variables, these are the features of this study. Subsequently, all independent variables are transformed into numbers.

Therefore, to optimize the discussion tree, other variables are required. Gini impurity is used to split the nodes, while the decision tree operates by splitting the example into portion groups, with at least one group used for testing and the remainder for training. Thus, the data frame is split into 70% for the training dataset and 30% for the test dataset. The decision tree operates at least five iterations. The accuracy and confusion aspects will be shown later .

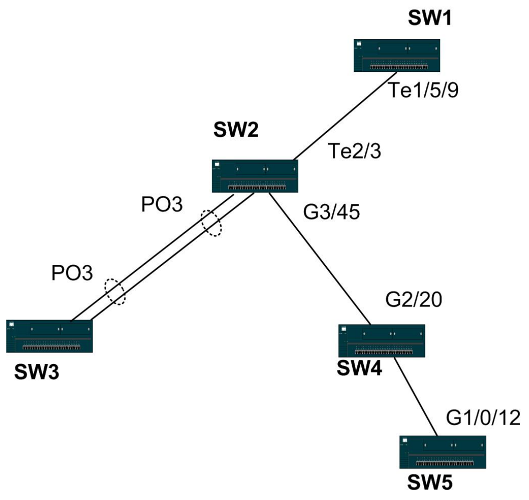

Fig. 7: *MFU network simulation.*

Briefly, this study uses SSL to perform data preparation, and then a decision tree to construct the best model.

4. PERFORMANCE EVALUATION

This section reveals the outcomes and results of the simulation and proposed network. The performance of this network is also compared to real-world networks.

4.1 Simulation

The simulation in this study is subject to the Mea Fah Luang University (MFU) network structure, which is composed of five main core switches as shown in Fig. 7. The network simulation involves the Cisco catalyst 9500 48 ports (SW1 and SW2), Cisco catalyst 6800 48 ports (SW4), Cisco catalyst 6500 48 ports (SW4), and Cisco 2960 24 ports (SW5). The connection between SW1 and SW2 is 10 Gbps, while the connection of each remaining switch is 1 Gbps. The LES retrieves the VLAN and MAC addresses from the MFU switch, comprising 159 VLANs and 4216 MAC addresses. The LES then randomly sets the probable attribute for these networks along with label I/N at the target, which is the predicting attribute. These examples are subsequently trained in the decision tree. Two decision trees are compared in this study: (1) Rapidminer and (2) Scikit-learn (sklearn) machine learning. Five iterations and five examples are used for sklearn, and 10 iterations and 10 examples for Rapidminer. Finally, the LES is compared with other machine learning algorithms.

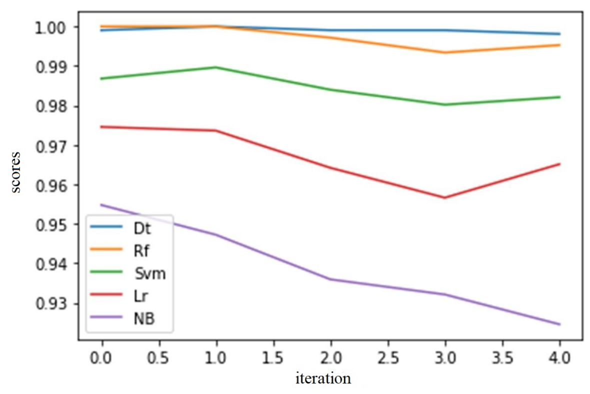

Fig. 8: *Cross validation.*

Table 2: *RapidMiner confusion matrix.*

Accuracy: 96.19% ±1.26% (micro average: 96.19%)

	True I	True N	Class Precision
Predict I	4224	147	96.64%
Predict N	190	4278	95.75%
Class recall	95.70%	96.68%	

4.2 Results and Performance Evaluation

In this section, the performance of the proposed model is evaluated using the SSL algorithm to create the training set. The accuracy of RapidMiner is 96.19% when using 8836 samples and 10 iterations, as indicated by the confusion matrix in Table 2. Whereas, predicted as N actually I is 190, predicted as I actually N is 147. Thus, in terms of true/false-positives and true/false-negative rates, the abnormal success rate is 96.64% with a false-positive rate of 3.36%, whereas the normal success rate is 95.75% with a false-negative rate of 4.25%. Whereas, the accuracy of sklearn is 99% when 8840 samples are used. Furthermore, the sklearn machine learning algorithm is also compared with other machine learning methods using the cross-validation technique to identify which of the aforementioned techniques is best for solving the problem under study. The results demonstrate that the decision tree has the best accuracy (approximately 99% of each iteration), as shown in Fig. 8. It can be clearly observed that the decision tree (Dt) is the winner, followed by random forest (RF), support vector machine (Svm), logistic regression (Lr), and the naïve Bayes (NB) algorithm, respectively. The mean scores are shown in Fig. 9.

Compared to other protocols such as STP and MAC address table learning, the proposed approach exhibits 1.2% fewer errors.

Additionally, the model in this study is applied to the MFU backbone. The results indicate that, out of 4802 MAC addresses, 4744 are N, and 58 are I. In other words, 1.2% of MAC addresses are abnormal. Overall, the MFU network remains in a normal situation even though the performance

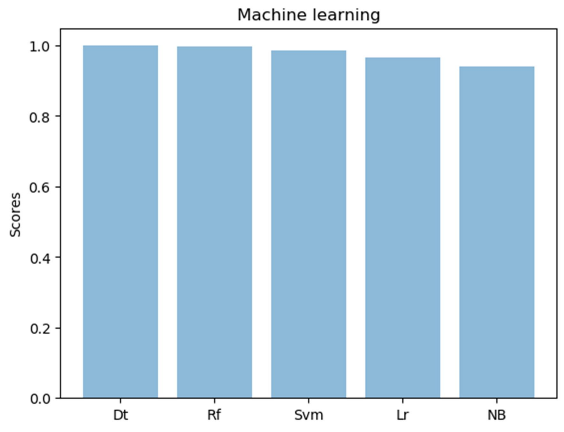

Fig. 9: *Machine learning scores.*

indicates a slowdown. Therefore, these issues might occur as a result of others.

5. DISCUSSION AND CONCLUSION

This work emphasizes the traditional layer-2 network evaluation by indicating an irregular MAC address in which machine learning is used. This paper proposes a method for evaluating the setup paths of STP and MAC address learning. The proposed algorithm is known as SSL. It prepares the data and then labels the prediction before applying it to machine learning. A comparison of each individual machine learning algorithm reveals that the proposed algorithm, which uses a decision tree, has the highest accuracy and confidence level.

To enable accurate forwarding, the STP is known to be the most appropriate algorithm for creating the instance of each path in each VLAN to alleviate the loop in the network. Thereafter, the MAC address table learns through flood and learning each instance. Neither of the algorithms can evaluate the entire layer-2 network path. According to the LES, MAC address path evaluation is applicable. The findings reveal that the SSL with a decision tree has the highest accuracy (99%) when using 8836 samples, with a 1.2% error in the existing protocols. Thus, when applying the proposed method to a real network, sklearn (an open-source ML Python library) is appropriate for SSL. The limitations of this study concern the number of MAC addresses, because using a large number of MAC addresses is very time consuming. Overall, the proposed method is extremely useful for network administrators in managing and monitoring their networks. It can also be applied to the Mea Fah Luang University network (to monitor, evaluate, and improve network performance). The proposed algorithm is also extremely useful for evaluating the layer-2 backbone anomaly and can be applied to new modern network fabrics.

Future studies should involve further complex algorithms and the investigation of other factors to optimize the problems and classify any other issues, including those concerning the new modern network fabric.

ACKNOWLEDGMENT

The author would like to thank the Center for Information Technology Service of the MFU and Technological Research and Modernizing Academic Learning (CENTRAL), which is funded by the Erasmus+ program, for their support and enabling the retrieval of vital information.

REFERENCES

[1] H. B. Acharya, J. Hamilton, and N. Shenoy, "From Spanning Trees to Meshed Trees," in *2020 International Conference on COMmunication Systems & NETworkS (COMSNETS)*, Jan. 2020, pp. 391–395, doi: 10.1109/COMSNETS48256.2020.9027495.

[2] P. Ayegba, J. Ayoola, E. Asani, and A. Okeyinka, "A Comparative Study Of Minimal Spanning Tree Algorithms," in *2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS)*, Mar. 2020, doi: 10.1109/ICMCECS47690.2020.240900.

[3] P. Willis and N. Shenoy, "A Meshed Tree Protocol for Loop Avoidance in Switched Networks," in *2019 International Conference on Computing, Networking and Communications (ICNC)*, Feb. 2019, pp. 303–307, doi: 10.1109/ICNC.2019.8685664.

[4] B. Hamid, Q. Rouland, and J. Jaskolka, "Distributed Maintenance of a Spanning Tree of k-Connected Graphs," in *2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC)*, Dec. 2019, pp. 217–226, doi: 10.1109/PRDC47002.2019.00052.

[5] J. Liu and Q. Xu, "Machine Learning in Software Defined Network," in *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, Mar. 2019, pp. 1114–1120, doi: 10.1109/ITNEC.2019.8729331.

[6] J. Wu, Y. Peng, M. Song, M. Cui, and L. Zhang, "Link Congestion Prediction using Machine Learning for Software-Defined-Network Data Plane," in *2019 International Conference on Computer, Information and Telecommunication Systems (CITS)*, Aug. 2019, doi: 10.1109/CITS.2019.8862098.

[7] A. Khan, A. A. Aesha, and J. Sarker, "A New Algorithmic Approach to Finding Minimum Spanning Tree," in *2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT)*, Sep. 2018, pp. 590–594, doi: 10.1109/CEEICT.2018.8628095.

[8] D. Rafique and L. Velasco, "Machine Learning for Network Automation: Overview, Architecture, and Applications [Invited Tutorial]," *Journal of Optical Communications and Networking*, vol. 10, no. 10, pp. D126–D143, Oct. 2018, doi: 10.1364/JOCN.10.00D126.

[9] D. Côté, "Using Machine Learning in Communication Networks [Invited]," *Journal of Optical Communications and Networking*, vol. 10, no. 10, pp. D100–D109, Oct. 2018, doi: 10.1364/JOCN.10.00D100.

[10] T. Anusas-Amornkul, "A Network Root Cause Analysis and Repair System," in *2018 6th International Symposium on Computational and Business Intelligence (ISCBI)*, Aug. 2018, pp. 69–73, doi: 10.1109/ISCBI.2018.00023.

[11] Z. Yang and K. L. Yeung, "An Efficient Algorithm for Constructing Controller Trees in SDN," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Dec. 2017, doi: 10.1109/GLOCOM.2017.8254747.

[12] S. Okada, T. Matsukawa, T. Tojo, S. Arai, and S. Yasukawa, "Reliability evaluation of fabric architecture as a replacement for router in carrier network," in *2017 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, Jun. 2017, doi: 10.1109/LANMAN.2017.7972152.

[13] W. Hou, L. Shi, Y. Wang, F. Wang, H. Lyu, and M. St-Hilaire, "An improved SDN-based fabric for flexible data center networks," in *2017 International Conference on Computing, Networking and Communications (ICNC)*, Jan. 2017, pp. 432–436, doi: 10.1109/ICNC.2017.7876167.

[14] P.-W. Chi, M.-H. Wang, J.-W. Guo, and C.-L. Lei, "SDN Migration: An Efficient Approach to Integrate OpenFlow Networks with STP-Enabled Networks," in *2016 International Computer Symposium (ICS)*, Dec. 2016, pp. 148–153, doi: 10.1109/ICS.2016.0038.

[15] S.-Y. Wang, C.-C. Wu, and C.-L. Chou, "Constructing an optimal spanning tree over a hybrid network with SDN and legacy switches," in *2015 IEEE Symposium on Computers and Communication (ISCC)*, Jul. 2015, pp. 502–507, doi: 10.1109/ISCC.2015.7405564.

[16] S. Rai and S. Sharma, "Determining minimum spanning tree in an undirected weighted graph," in *2015 International Conference on Advances in Computer Engineering and Applications*, Mar. 2015, pp. 637–642, doi: 10.1109/ICACEA.2015.7164769.

[17] L. Yanjun, L. Xiaobo, and Y. Osamu, "Traffic engineering framework with machine learning based meta-layer in software-defined networks," in *2014 4th IEEE International Conference on Network Infrastructure and Digital Content*,

Sep. 2014, pp. 121–125, doi: 10.1109/ICNIDC.2014.7000278.

[18] J. Li and J. Zhu, "Network Expansion Problem on the Spanning Tree in Graphs," in *2009 WRI World Congress on Computer Science and Information Engineering*, Mar. 2009, vol. 2, pp. 691–695, doi: 10.1109/CSIE.2009.336.

[19] X. Xiao-Hua, N. Ai-Bing, and M. Liang, "A new verification algorithm for minmum spanning tree based on reduction and merge technology," in *2009 4th International Conference on Computer Science & Education*, Jul. 2009, pp. 469–473, doi: 10.1109/ICCSE.2009.5228388.

[20] M. A. Kashem, C. S. Hasan, and A. Bhattacharjee, "An Algorithm for Solving the Minimum Vertex-Ranking Spanning Tree Problem on Series-Parallel Graphs," in *2006 International Conference on Electrical and Computer Engineering*, Dec. 2006, pp. 328–332, doi: 10.1109/ICECE.2006.355638.

Mahamah Sebakor received his B.Eng. degree in computer engineering, M.Eng. degree in electrical engineering, and Ph.D. degree in electrical engineering from Chiang Mai University, Chiang Mai, Thailand in 1998, 2008, and 2016, respectively.

From 1998 to 2019, he was with Information Technology Service Center, Chiang Mai University, Chiang Mai, Thailand as a Senior Network Engineering. From 2018 to 2019, he was with the faculty of computer engineering, Chiang Mai University, Chiang Mai, Thailand as a special instructor. From 2017 to 2019, he was with the department of computer engineering, Chiang Mai Technical College, Chiang Mai, Thailand as a special instructor. He is currently working as a lecturer at the school of information technology, computer engineering, Mea Fah Luang University, Chiang Rai, Thailand. He also has the Cisco Certified Internetwork Expert (CCIE #22976). His research interest includes network, wireless network, cyber security, software defined, network programing, artificial intelligence, machine learning, and other network technologies.