International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING

ISSN:2147-6799 **www.ijisae.org** **Original Research Paper**

Assessment of Conflict Flows in Software-Defined Networks using a Novel Nature-Inspired Optimization-Tuned Kernelized SVM

Amit Sharma[1], Veena M.[2], Dr.Hirald Dwaraka Praveena[3], Dr. V. Selvakumar[4], Bhuvana J.[5], Dhiraj Singh[6]

Submitted: 07/02/2024 **Revised:** 15/03/2024 **Accepted:** 21/03/2024

Abstract*:* The centralizing management and flexibly customizing network resources, Software-Defined Networks (SDN) completely transform network administration. As networks become more intricate, the likelihood of conflicts arising data flows increases, potentially leading to a decline in overall performance and the emergence of security vulnerabilities. This paper presents a tree-seed optimization-tuned kernelized support vector machine (TSO-KSVM) for the assessment of conflict flows in SDN environments. Initially, we gather data samples of SDN in conflict flows to analyze the performance of the proposed method. Applying the min-max scaling method to preprocess the raw data samples and linear discriminant analysis (LDA) is carried out to reduce the dimension. In the proposed framework, TSO is applied to enhance the assessment in the KSVM model. The proposed method is implemented in the Python tool. The proposed method's performance is analyzed in terms of various metrics compared with other methods. From the experimented results, we conclude that the proposed method attains the greatest accuracy rate of other methods in assessing conflict flows in SDN networks.

Keywords: *Conflict Flow, Security, Software-Defined Network (SDN), Tree-Seed Optimization-Tuned Kernelized Support Vector Machine (TSO-KSVM)*

1. Introduction

Network performance can be enhanced through Software-Defined Networks (SDNs) using dynamic and adaptable network architecture. To accommodate changing business requirements, network engineers and administrators can easily modify this architecture using a centralized management center [1]. Additionally, it makes possible for network engineers and managers to adapt the changing business demands through a centralized management dashboard. SDNs combine network technologies and detach control functions from forwarding planes to provide agility and flexibility [2]. System experts can have total control over the network's operations due to this divide, which might enable a separate network control plane setup. Aside from its core benefits, an SDN is economical, dynamic, controlled, adaptive and adaptable, which makes an ideal answer to the growing size and high-bandwidth problems and internet-based apps [3]. A range of network technologies are included in the SDN, which are intended to make the network strong and scalable enough to support virtualized servers and storage infrastructures in a contemporary data center [4]. SDNs are well suited for the dynamic demands of modern high-bandwidth applications since they are inexpensive, controlled, dynamic and adaptable [5]. Network control operations are isolated from the original network sending traffic in reverse by SDN's virtualized execution architecture. Apart from the integration of diverse network devices such as switches, routers and access points in SDN, which facilitates the execution of multiple network management operations, the SDN controller permits intricate network setup [6]. Essentially, the basic giving consumers greater choice over their setup while maintaining network efficiency standards is the aim of SDN. One significant development in SDN is conflict flow [7]. Flow conflicts can take many various shapes, such as in the case of SDN design. There can be a variety of conflicts between the controller and flow table due to variations in the flow rule policy or flow entry. An SDN's success is influenced by the controller's actions [8]. To distribute, categorize and allocate packets depending on flow entries, a conflict flow switch contains many flow tables connected to the controller through the conflict flow protocol identified, in this study uses machine learning (ML) techniques for SDN.

- To assess the network administration is revolutionized by SDN, which centralizes control and allows for flexible customization of network resources.
- For the assessment of conflict flows in SDN environments, in this study we employed a TSO-KSVM.
- The evaluation of the suggested algorithms' performance using criteria such as precision, accuracy, recall, False alarm rate and F1-score.

[1] *Professor School of Computer Applications Lovely Professional University, Phagwara, Punja, India. profamitsharma@gmail.com*
[2] *Assistant Professor, Department of Computer science and Engineering, PES College of engineering, Mandya, Karnataka, India. veenakemps@gmail.com*
[3] *Assistant Professor, Department of ECE, School of Engineering, Mohan Babu University (Erstwhile Sree Vidya nikethan Engineering College) Tirupati, Andhra Pradesh, India. hdpraveena@gmail.com*
[4] *Assistant Professor, Maths and Statistics Bhavan's Vivekananda College of Science, Humanities and Commerce Hyderabad-94, Telangana, India. drselva2022@gmail.com*
[5] *Associate Professor, Department of Computer Science and Information Technology, Jain (deemed to be University), Bangalore, Karnataka, India. j.bhuvana@jainuniversity.ac.in*
[6] *Centre of Research Impact and Outcome, Chitkara University, Rajpura, Punjab, India. dhiraj.singh.orp@chitkara.edu.in*

The rest of the paper is divided into parts: The objective is based on the related studies presented in part 2. The data collection and their proposed techniques are shown in part 3. The result analysis and their discussion are shown in parts 4 and 5. The end of the paper was concluded in section 6.

2. Related Works

The study [9] suggested a unique flow schedule-generating model that adds no additional update cost and ensures no frame loss during network updates, even with the most basic two-phase update method**.** The study [10] suggested the method reveals significant barriers. By examining network behavior with many applications and traffic profiles applied to various topologies, the experimental technique generates conflict classes and detection patterns. The study [11] provided the flow conflicts in SDN by identifying and categorizing every flow conflict in the Open Flow switch using ML methods. The study [12] suggested in an attempt to address the problem. Initially, the information on the flow rules to be issued by watching and recording Open Flow messages amid the forwarding plane and the control plane. The study [13] suggested the novel networking design called SDN separates the administrative and control planes from the data plane of forwarding devices. The controller is a centralized entity that implements the management and organization planes. The study [14] focused on the identification of conflicts in SDN.The flexible deployment of network functionalities was made easier by the SDN architecture. Compared to traditional networks, this design increases the likelihood of disputes while encouraging innovation. The study [15] examined the separating network operations from devices and centralizing them at a logical location so-called SDN controller while keeping a shared communication interface, the SDN architecture makes easier to install network services flexibly. The study [16] examined the SDN, which was extensively utilized in many different settings allows for flexible network configuration can identify and resolve conflicts between flow entries using the approach suggested in the work. The study [17] addressed the problem of detecting situations when the SDN stack was unable to stop these components' policies from becoming inconsistent. The study [18] provided a hybrid SDN flow control method based on clusters and the method leverages both centralized SDN routing and distributed legacy routing, making it hybrid.

2.1. Problem statement

Efficiency is negatively impacted by many conflict types in both SDNs and traditional networks. Based on their guidelines and outcomes, interpretative along with intelligible conflicts are the two primary categories of disputes. When resolving flow conflicts, packet counts and timeout numbers are not crucial. Action and priority modifications are two examples of qualities that might lead to conflicts in SDN. The controller and flow table can become incompatible with modifications made to the flow rule policies or entries. To create SDN rules and flow entries must decide and take action. It is acknowledged that traditional networks and SDNs have quite different qualities, especially when it comes to priority and action. By dynamically arranging behaviors, intelligently managing flow entry constraints and adjusting to the unique characteristics of both types of connections, the TSO-KSVM method effectively resolves conflicts in both SDNs and standard networks, providing optimum efficiency and solving conceptual and comprehensible conflicts.

3. Methods

This study provides further details on the conflict flow in SDN that was utilized for this work to support the suggested TSO-KSVM-based SDN architecture. Fig 1 shows the suggested methodology.

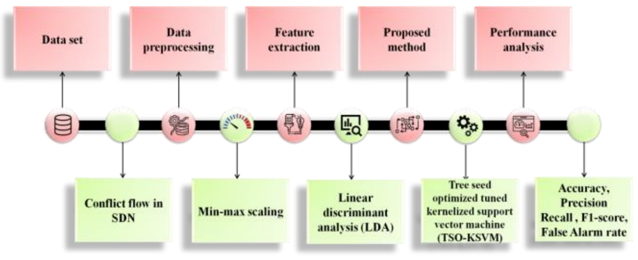
Data set
Data preprocessing
Feature extraction
Proposed method
Performance analysis
Conflict flow in SDN
Min-max scaling
Linear discriminant analysis (LDA)
Tree seed optimized tuned kernelized support vector machine (TSO-KSVM)
Accuracy, Precision Recall , F1-score, False Alarm rate

Fig.1 Proposed Flow

3.1. Dataset

The collection of data that utilized Ryu controller architecture and 10 performance servers per host, Iperf is a program that creates and gathers Open Flow data [19]. Different destination ports can be listened by each server and new flows are created, depending on the source and destination ports and protocols, in addition to the source and destination IP addresses. The controller is in charge of adding new flows to the switch, revising policy guidelines and compiling all flow inputs into a CSV file. An SDN controller can build flows using a new way suggested by the study. The absence of SDN datasets with crucial flow entry attributes like priority and action elements made this essential. The network flows that the switch has detected and handled are represented by these flow entries.

3.2. Data preprocessing using Min-Max scaling

There can be multiple values and possibly missing values in the SDN data. An error is produced at the end of the assessment if there are any missing or duplicate values in the dataset. Equation (1) provides the mean of the estimation.

$$K_i = \frac{K_{i+1}+K_{i-1}}{2} \quad (1)$$

Where following missing value is indicated as K_{i+1}, whereas the value before shown as K_{i-1}.The Min-Max scalar function is used to remove redundant data. This scaling strategy allows features in an SDN dataset to have similar scales, preventing specific features from dictating how the ML models learn. Equation (2) is the fundamental formula for applying Min-Max scaling to numerical data that has to be scaled withthe framework of a monitoring and control system.

$$Z_{scaled} = \frac{z-z_{Min}}{z_{Min}-z_{Max}} \quad (2)$$

Where scaling values of the source feature are represented by Z_{scaled}. The feature's initial value is represented by Z. The SDN dataset's feature's minimum value is denoted byZ_{Min}. The highest possible value of the feature in the dataset is represented by Z_{Max}.

3.3. Feature Extraction using Linear Discriminant Analysis (LDA)

Linear discriminant analysis (LDA) is increasingly used to extract critical information from SDN data and reduce the dimensionality of data. LDA can be influenced by several factors operating simultaneously. Multi-feature prediction issue is well-suited to dimensionality reduction, the central idea of LDA, which seeks to convert the transformation of high-dimensional data into a pleasing low-dimensional form. Specifically, suppose Y_j, μ_j, Σ_j indicating $j-th$ the data set, the average, and the matrix of correlations for the sample. $\omega^T\mu_1$and$\omega^T\mu_2$ entails projecting the middle of the samples $J\epsilon\ \{1,2\}$ on line ω whereas $\omega^T\Sigma_{1\omega}$ and $\omega^T\Sigma_{2\omega}$ represent the matrix of the two samples' covariance. Distance between homogenous sample points is the target of LDA, i.e.$\omega^T\Sigma_{1\omega} + \omega^T\Sigma_{2\omega}$ minimize the distance between points in the sample that are diverse, i.e. $\|\omega^T\mu_1 - .\ \omega^T\mu_2\|_2^2$ 2 over an extended period. Here provide the objective function$I$as in equation (3)

$$J = \frac{\|\omega^T\mu_1 - \omega^T\mu_2\|_2^2}{\omega^T\Sigma_{1\omega} + \omega^T\Sigma_{2\omega}} = \frac{\omega^T(\mu_1-\mu_2)(\mu_1-\mu_2)^T\omega}{\omega^T(\Sigma_1+\Sigma_2)\omega} \quad (3)$$

This is the definition of the inner-class divergence matrix in equation (4)

$$S_W = \Sigma_1 + \Sigma_2 = \sum_{y\in Y_1}(Y-\mu_1)(Y-\mu_2)^T + \sum_{y\in Y_2}(Y-\mu_1)(y-\mu_2)^T \quad (4)$$

Thus the matrix of divergence between classes can be written as equation (5)

$$S_b = (\mu_1-\mu_2)(\mu_1-\mu_2)^T \quad (5)$$

Consequently, (3) is reduced to equation (6)

$$J = \frac{\omega^T S_b\omega}{\omega^T S_W\omega} \quad (6)$$

This ωmatrix is solved using the Lagrangian multiplier technique and the singular value decomposition (SVD) method $\omega = S_w^{-1} = (\mu_1-\mu_2)$ i.e. the optimal direction for projection. The next step is to determine the optimal course of action using the training data$\omega asZ = \omega^T Y$.

$$\begin{cases} Z > y_0 \Rightarrow y\epsilon Class_1 \\ Z < y_0 \Rightarrow y\epsilon Class_2 \end{cases} \quad (7)$$

$$z_0 = \frac{N_1\mu_1 + N_2\mu_2}{N_1+N_2} \quad (8)$$

Equations (7) and (8) determine the discrimination outcome of $N_1 and N_2$ to show the amounts of actual and simulated collisions.

3.4. Conflict flow in SDN environment using TSO-KSVM

The tree-seed optimization-tuned kernelized support vector machine (TSO-KSVM), which optimizes flow management for effective and safe data transfer, improves conflict detection in an SDN context with the ability to adjust the changes in the network, guaranteeing dependable and effective communication. This mixture improves classification precision and smoothly resolves disagreements.

3.4.1. Kernelized Support Vector Machine (KSVM)

The kernelized Support Vector Machine (KSVM) is a statistical learning theory-based tool for classifying data that is linear and nonlinear. It creates a linear optimum splitting hyperplane with a higher dimension (or classes) by separating the data into two groups using margins and support vectors. The initial training data is transferred using a suitable non-linear mapping. In the design, "the eigenvectory_j, the kernel function K, the output vector$U$, and the bias term c" are all shown. The following describes the model's final decision function in equation (9).

$$f(U) = sgn\left(\sum_{j=1}^{s} Y_j L(u, U_j) + c\right) \quad (9)$$

Equation (10) shows that the highest predictive power is provided by the radial basis function. It requires absolute improvements in time spent, mistakes or fit quality.

$$R(u_o - e_j) = exp\left(-\frac{1}{2\sigma^2}u_o - e_j^2\right), \quad (10)$$

Where$u_o - e_j$is the Euclidean norm, $e_j$is the center of the Gaussian function and its variance. The output of the neural network structure supplied with the radial basis function is follows:

$$vj = \sum_{j=1}^{g} y_{ji} exp\left(-\frac{1}{2\sigma^2}u_o - e_j^2\right), i = 1,2,\ldots s \quad (11)$$

Where$u_o = (u_1^o, u_2^o, \ldots u_n^o)^S$is the input sample $(o = 1,2,3,\ldots,O)$ is the overall count of pieces, $x_{ji}$is the weight of the connection$(j = 1,2,3,\ldots,g)$, where g is the number of nodes. The variance of the basis function can be defined as follows using the least squares method:

$$\sigma = \frac{1}{O} = \sum_i^n c_i - z_j\, e_j^2 \quad (12)$$

Typically, the default value of the parameter in the gamma function is set to the opposite of the attribute count. Equation (11) has the potential to develop into a new function for making decisions based on Equations (12) and (13).

$$f(u)sgn\left(\sum_{j=1}^{m} y_j exp\left(-gammay_j - u^2\right) + b\right), \quad (13)$$

To make sure the model behaves as intended, the gamma term and penalty term of the KSVM must also be supplied. As the penalty term value decreases, the probability of under fitting rises, whereas the probability of overfitting increases. If the punishment term's value is too high or low, it will hinder the KSVM's ability to generalize. Moreover, research ignores the importance of the gamma value of the KSVM. Algorithm 1 shows the KSVM Algorithm.

Algorithm 1: KSVM

```
$Require: W\ and\ z\ loaded\ with\ training\ labeled\ data, \alpha$
        $\Leftarrow 0\ or\ \alpha \Leftarrow partially\ trained\ KSVM$
1: $D \Leftarrow some\ value\ (10\ for\ example)$
2: $repeat$
3: $for\ all\ \{w_j, z_j\}, \{w_j, z_j\}, do$
4: $optimize\ \alpha_j\ and\ \alpha_i$
5: $End\ for$
6: $until\ no\ changes\ in\ \alpha\ or\ another\ resource\ constraint$
$criterial\ met\ Ensure$:
 $Retain\ only\ the\ support\ vectors\ (\alpha_j > 0)$
```

3.4.2. Tree-Seed Optimization

Discovering the ideal answer to an issue becomes more challenging as its complexity rises. Therefore, we have to change parts of the original algorithms' structures to get efficient results. When the problem's dimension rises, it becomes more difficult to solve optimally using the original Tree-Seed Optimization (TSO). Because of this, the TSO has undergone several modifications to enhance its operations. The original TSO had the acceleration coefficient C parameter, which is determined andbased on the problem's dimensions. The novel equations employed in this work are (14) and (15), which are shown below.

$$\Delta_{j,i} = \left(BestTree_i - Trees_{q,i}\right) * (rand - 0.5) * D \quad (14)$$

$$\Delta_{j,i} = \left(ParentTree_i - Trees_{q,i}\right) * (rand - 0.5) * D \quad (15)$$

Equation (16) is used to obtain the C parameter that is utilized in equations (14) and (15). The size of the issue is represented by the D parameter in Equation (16).

$$D = 2 - (C^2 * 0.0001) \quad (16)$$

Equation (17) provides the upper and lower limits for $\Delta_{j,i}$.

$$min \le \Delta_{j,i} \le Max \quad (17)$$

Δi, minimum and maximum values are computed successively $= -0.1 * (c_{max} - c_{min}) and min = -0.1 * (c_{max} - c_{min}).c_{min}$. The problem's top limit is represented by $dmax$, while the lower bound is represented by $dmin$. Then, for the seed location, $\Delta_{j,i}$is added to the current parent tree. Equation (18) shows a seed's location.

$$Seed_{j,i} = ParentTree_i + \Delta_{j,i} \quad (18)$$

4. Results

The experimental setup with 16 GB of RAM, an Intel Core i7 CPU, and Ubuntu 18.04 running, the user's system makes use of the Ryu SDN controller, Python 2.7 for scripting, and OpenFlow Switch Version 1.3 or above. The research is verified based on five distinct metrics that consist of False alarm rate (FAR), F1-measure, Accuracy, Precision, and Recall. Assessing the conflict flow for SDN by utilizing TSO-KSVM methods was the focus of this study we compared our suggested technique to Decision Tree (DT) [20], Deep Neural network (DNN) [20], K-nearest neighbor (KNN) [20] and Multi-level hybrid classification (MLHC) [20].

Accuracy: The proportion of accurately predicted occurrences to all instances is known as accuracy. SDN improves result accuracy by streamlining network performance. Conflicts in flow control might occur and reduce effectiveness.

Precision: Precision is defined as the ratio of predicted positive observations to the total number of anticipated positives. The precision of results is improved by SDN, which minimizes conflicts by controlling network traffic. Table 1 and Fig 2 provide the accuracy and precision comparison. In contrast to other current approaches, our suggested strategy produced superior results with 97.12% accuracy and 96.15% precision.

Table 1 Comparison of Accuracy and Precision

Methods	Accuracy (%)	Precision (%)
DT [20]	74.43	92.5
DNN [20]	75.75	83
KNN [20]	77.09	77.09
MLHC [20]	84.29	94.18
TSO-KSVM [Proposed]	97.12	96.15

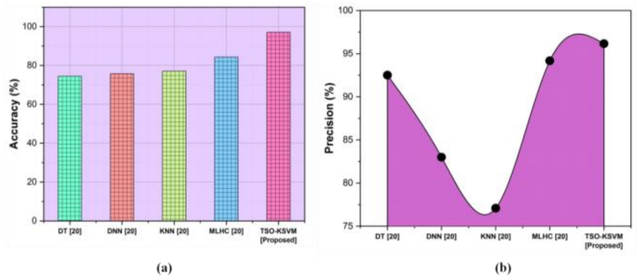

Fig.2 Outcome of (a) Accuracy, (b) Precision

Recall: The ratio of all observations in the actual class to the anticipated positive observations is known as recall. In SDN, rerouting traffic, enforcing policy and resolving conflicts are all part of result recall. By using historical results to settle disputes and maintain reliable flow control, it guarantees effective network functioning.

F1-score: The F1-score represents the balanced average of recall and accuracy. When the distribution of classes is not uniform, it is helpful and current initiatives to enhance flow management for better performance and aligning F1-scores in contexts with software-defined networking. A comparison of recall and F1-score is shown in Table 2 and Fig 3. Compared to other existing methodologies, our proposed methodology produced better outcomes with 92.1% of recall and 90.2% of F1-score.

Table 2 Comparison of Recall and F1-score

Methods	Recall (%)	F1-score (%)
DT [20]	59.95	72.75
DNN [20]	76	75
KNN [20]	62.84	75.75
MLHC [20]	77.18	84.83
TSO-KSVM [Proposed]	92.1	90.2

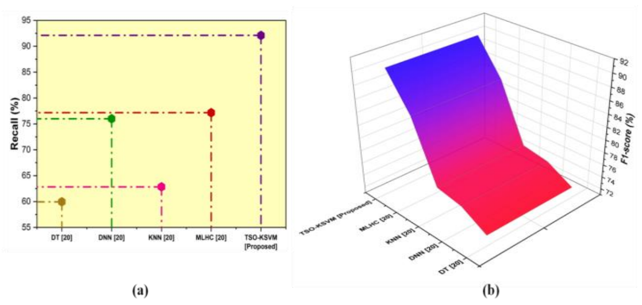

Fig.3 Outcome of (a) Recall, (b) F1-score

False alarm rate (FAR): The ratio of falsely projected positive observations to all real negative observations is known as the FAR. Table 3 and Fig 4 show the comparison of False Alarm Rate. Compared to other existing methods, our suggested techniques provide a lower FAR percentage of 2.31.

Table 3 Comparison of FAR

Methods	False Alarm Rate (%)
DT [20]	6.43
DNN [20]	3.21
KNN [20]	4.07
MLHC [20]	6.3
TSO-KSVM [Proposed]	2.31

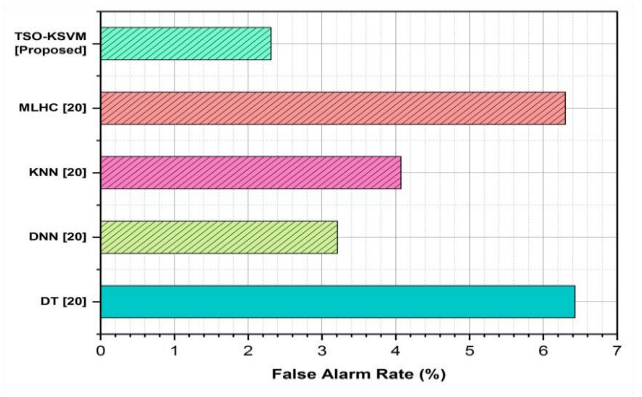

Fig.4 Outcome of FAR

5. Discussion

The experimental investigation made clear that traditional ML techniques, such as DT [20], KNN [20] and DNN [20], would not always provide the best outcomes. It was recommended to investigate other approaches, especially multi-level systems to improve performance. The research also highlighted the difficulties in tackling problems like conflict flow, emphasizing those findings utilizing ML techniques for the MLHC [20] method could not always transfer to effective detection of unknown threats in SDN. To summarize, the study recommends exploring various approaches, including multi-level systems, to get the intended outcomes and emphasizes the significance of complexity of the context, especially in SDN contexts, our proposed TSO-KSVM system shows that fewer false alarms are generated, which can be a consequence of our recommended techniques for attaining high accuracy through a Conflict flow of level in SDN.

6. Conclusion

To discover and categorize SDN model in conflict flows inside, the research offers ML techniques. By using the protocol, IP source address, flow rules' priorityand action, the various conflict kinds are identified and grouped. Increased network complexity increases the likelihood of flow collisions, which can lead to poor performance and security vulnerabilities. In this research, a new kernelized support vector machine (TSO-KSVM) optimized via tree-seed optimization is presented for the evaluation of conflict flows in SDN systems. The proposed TSO-KSVM method achieved the greater outcomes of precision 96.15%, accuracy 97.12%, f1-score 90.2%, recall 92.1%, and FAR 2.31%. Significant improvements in the identification and categorization of conflict fluxes inside SDN have been shown by the suggested method. This study is the first effort, to identify and categorize conflict flows using ML methods. Using the same dataset, future research will concentrate on investigating other ML techniques for conflict flow detection and classification.

References

[1]. LUDWIG–MAXIMILIANS–UNIVERSIT, A. M. Experimental Examination of Distributed Conflicts in Software Defined Networks.

[2]. Kafetzis, D., Vassilaras, S., Vardoulias, G., &Koutsopoulos, I. (2022). Software-defined networking meets software-defined radio in mobile ad hoc networks: state of the art and future directions. IEEE Access, 10, 9989-10014.

[3]. Guidara, A., Pomares Hernandez, S. E., Rodriguez Henriquez, L. M. X., HadjKacem, H., &HadjKacem, A. (2020). Towards causal consistent updates in software-defined networks. Applied Sciences, 10(6), 2081.

[4]. Khanmirza, H. (2022). WildMinnie: compression of software-defined networking (SDN) rules with wildcard patterns. PeerJ Computer Science, 8, e809.

[5]. Häckel, T., Meyer, P., Korf, F., & Schmidt, T. C. (2022). Secure time-sensitive software-defined networking in vehicles. IEEE Transactions on Vehicular Technology, 72(1), 35-51.

[6]. Tan, E., Chong, Y., & Anbar, M. F. (2022). Flow management mechanism in software-defined network. Comput. Mater. Cont, 1(70), 1437-1459.

[7]. Ujcich, B. E., Jero, S., Skowyra, R., Gomez, S. R., Bates, A., Sanders, W. H., &Okhravi, H. (2020, January). Automated discovery of cross-plane event-based vulnerabilities in software-defined networking. In Network and Distributed System Security Symposium.

[8]. Farooq, M. S., Riaz, S., &Alvi, A. (2023). Security and Privacy Issues in Software-Defined Networking (SDN): A Systematic Literature Review. Electronics, 12(14), 3077.

[9]. Pang, Z., Huang, X., Li, Z., Zhang, S., Xu, Y., Wan, H., & Zhao, X. (2020). Flow scheduling for conflict-free network updates in time-sensitive software-defined networks. IEEE Transactions on Industrial Informatics, 17(3), 1668-1678.

[10]. Tran, C. N., &Danciu, V. (2020). A general approach to conflict detection in software-defined networks. SN Computer Science, 1, 1-14.

[11]. Khairi, M. H. H. (2021). Flow Conflict Eliminations through Machine Learning for Software Defined Network (Doctoral dissertation, Ph. D. dissertation, UniversitiTeknologi Malaysia).

[12]. ZHANG, L., LIN, H., HUAN, W., & BI, W. (2022). Software defined network flow rule conflict detection system based on OpenFlow. Journal of Computer Applications, 42(2), 528.

[13]. Asif, A. B., Imran, M., Shah, N., Afzal, M., &Khurshid, H. (2021). ROCA: Auto-resolving overlapping and conflicts in Access Control List policies for Software Defined Networking. International Journal of Communication Systems, 34(9), e4815.

[14]. Tran, C. N. (2022). Conflict detection in software-defined networks (Doctoral dissertation, lmu).

[15]. Danciu, V., & Tran, C. N. (2020). Side-effects causing hidden conflicts in software-defined networks. SN Computer Science, 1(5), 278.

[16]. Tang, L., Fu, Y., Zeng, Y., Li, Z., & Li, S. (2021). Flow entry conflict detection and resolution scheme for software-defined networking. The International Journal of Electrical Engineering & Education, 0020720921998237.

[17]. Lee, S., Woo, S., Kim, J., Yegneswaran, V., Porras, P., & Shin, S. (2020, July). AudiSDN: Automated detection of network policy inconsistencies in software-defined networks. In IEEE INFOCOM 2020-IEEE Conference on Computer Communications (pp. 1788-1797). IEEE.

[18]. Liu, Q., Cheng, L., Alves, R., Ozcelebi, T., Kuipers, F., Xu, G., ...& Chen, S. (2021). Cluster-based flow control in hybrid software-defined wireless sensor networks. Computer Networks, 187, 107788.

[19]. Khairi, M. H., Abdalla, B. M. A., Hassan, M. K., Ariffin, S. H., &Hamdan, M. (2024). Utilizing Extremely Fast Decision Tree (EFDT) Algorithm to Categorize Conflict Flow on a Software-Defined Network (SDN) Controller. Engineering, Technology & Applied Science Research, 14(2), 13261-13265.

[20]. Latah, M., &Toker, L. (2020). An efficient flow-based multi-level hybrid intrusion detection system for software-defined networks. CCF Transactions on Networking, 3(3-4), 261-271.