IEEE Access
Multidisciplinary ⁝ Rapid Review ⁝ Open Access Journal

SPECIAL SECTION ON INTELLIGENT BIG DATA ANALYTICS FOR INTERNET OF THINGS, SERVICES AND PEOPLE

Received May 1, 2021, accepted May 10, 2021, date of publication May 17, 2021, date of current version May 28, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3081629

Detection and Classification of Conflict Flows in SDN Using Machine Learning Algorithms

MUTAZ HAMED HUSSIEN KHAIRI[1,2],
SHARIFAH HAFIZAH SYED ARIFFIN[1], (Senior Member, IEEE),
NURUL MU'AZZAH ABDUL LATIFF[1], (Senior Member, IEEE),
KAMALUDIN MOHAMAD YUSOF[1], MOHAMED KHALAFALLA HASSAN[1,2],
FAHAD TAHA AL-DHIEF[1], (Graduate Student Member, IEEE),
MOSAB HAMDAN[1] (Graduate Student Member, IEEE),
SULEMAN KHAN[3], AND MUZAFFAR HAMZAH[4]

[1]Faculty of Engineering, School of Electrical Engineering, Universiti Teknologi Malaysia (UTM), Johor Bahru 81310, Malaysia
[2]Faculty of Engineering, Future University, Khartoum 10553, Sudan
[3]Department of Computer and Information Sciences, Northumbria University, Newcastle Upon Tyne NE1 8ST, U.K.
[4]Faculty of Computing and Informatics, Universiti Malaysia Sabah, Kota Kinabalu 88400, Malaysia

Corresponding authors: Mutaz Hamed Hussien Khairi (taza1040@gmail.com), Sharifah Hafizah Syed Ariffin (shafizah@utm.my), and Muzaffar Hamzah (muzaffar@ums.edu.my)

This work was supported in part by the Research Team Computer Networks and System (CSNET), and in part by the Ministry of Education Malaysia (MOE) and Research Management Centre UTM (RMC).

ABSTRACT Software-Defined Networking (SDN) is a new type of technology that embraces high flexibility and adaptability. The applications in SDN have the ability to manage and control networks while ensuring load balancing, access control, and routing. These are considered the most significant benefits of SDN. However, SDN can be influenced by several types of conflicting flows which may lead to deterioration in network performance in terms of efficiency and optimisation. Besides, SDN conflicts occur due to the impact and adjustment of certain features such as priority and action. Moreover, applying machine learning algorithms in the identification and classification of conflicting flows has limitations. As a result, this paper presents several machine learning algorithms that include Decision Tree (DT), Support Vector Machine (SVM), Extremely Fast Decision Tree (EFDT) and Hybrid (DT-SVM) for detecting and classifying conflicting flows in SDNs. The EFDT and hybrid DT-SVM algorithms were designed and deployed based on DT and SVM algorithms to achieve improved performance. Using a range flows from 1000 to 100000 with an increment of 10000 flows per step in two network topologies namely, Fat Tree and Simple Tree Topologies, that were created using the Mininet simulator and connected to the Ryu controller, the performance of the proposed algorithms was evaluated for efficiency and effectiveness across a variety of evaluation metrics. The experimental results of the detection of conflict flows show that the DT and SVM algorithms achieve accuracies of 99.27% and 98.53% respectively while the EFDT and hybrid DT-SVM algorithms achieve respective accuracies of 99.49% and 99.27%. In addition, the proposed EFDT algorithm achieves 95.73% accuracy on the task of classification between conflict flow types. The proposed EFDT and hybrid DT-SVM algorithms show a high capability of SDN applications to offer fast detection and classification of conflict flows.

INDEX TERMS Software-defined network, conflict flows detection, flow classification, machine learning algorithms.

I. INTRODUCTION

The conventional architecture of a network is not fully adaptable to the requirements of using the current network applications and advanced data centre environments. Therefore, Software-Defined Network (SDN) was proposed to allow cloud and network administrators as well as engineers to of keep up with the ever-changing business requirements over a centralized control console [1]. The SDN also includes a variety of network technologies designed to make the

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Khurram Khan.

network scalable and robust enough to accommodate storage infrastructures and virtualized servers in a modern data centre. Furthermore, the SDN technique is originally intended for managing, constructing, and designing networks. This is to provide direct network programmability and independence of the primary infrastructure of network services and applications by separating the network control and forwarding planes. In general, the SDN is cost-effective, manageable, dynamic, and adaptable which makes it appropriate for the dynamic nature of high-bandwidth modern applications [2]. SDN presents a virtualised execution framework which separates the network control functions from the underlying network forwarding traffic [3]. In addition to the incorporation of various network devices in SDN (e.g., routers, switches, and access points) which allows for the implementation of various network control functions, the SDN controller allows complicated network configuration. In essence, the main goal of the SDN is to give users more control over their control configuration while still ensuring network efficiency requirements [4].

The reliability of a traditional network is known to be harmed by various types of conflicts to which SDN is not immune. Intelligible Conflicts and Interpretative Conflicts are the two major forms of conflict categories according to their laws and effects. Flow conflict manifest in different forms, such as the case of designing the SDN. Changes in the flow rule policy or flow entry result in different forms of conflict appearing in the controller and flow table. On the other hand, the techniques for detecting and classifying flow conflicts in SDN models are highly imperative. For example, Machine Learning (ML) algorithms have proved their efficiency and effectiveness in detecting and classifying two or more subjects [5], [6], with application in several domains such as identification of spam emails [7], images classification in the medical domain [8], [9], voice pathology detection [10]–[12], and language identification [13], [14]. In these methods, the implemented ML algorithms played the main role. The main purpose of using these algorithms is to train and build a system that is efficiently capable of classifying subjects with high detection accuracy. However, ML algorithms are still suffering from low detection accuracy in SDN models. Besides, these algorithms have received minimal attention with respect to the detection and classification of the flow conflicts in SDN compared to other domains. In other words, no work represents the detection and classification of flow conflict types using ML algorithms

This paper discusses the use of Decision Tree (DT), Support Vector Machine (SVM), Extremely Fast Decision Tree (EFDT), and Hybrid (DT-SVM) machine learning algorithms for detecting and classifying flow conflicts in SDNs. To improve the performance of the flow conflict detection system, the EFDT and hybrid DT-SVM algorithms were designed and deployed based on DT and SVM algorithms. A variety of evaluation metrics are used to assess the performance of the proposed algorithms in terms of efficiency and effectiveness. The following are the key contributions of this paper:

- Proposed Extremely Fast Decision Tree (EFDT) and hybrid Decision Tree-Support Vector Machine (DT-SVM) for flow conflict detection in SDN.
- Application of DT, SVM, EFDT, and DT-SVM ML algorithms in the detection and classification of conflict flows.
- The use of different number of flows with the algorithms each time to identify and classify conflict flows.
- The use of accuracy, precision, F1-score, recall, and execution time assessment metrics to assess the performance of the proposed algorithms.
- To the best of our knowledge, the is first attempt at using machine learning algorithms to identify and classify conflict flows.

The rest of this paper is organized as follow; Section II addresses similar works of ML algorithms used in the SDN domain. The suggested methods are presented in Section III. The experimental results and discussion are detailed in Section IV. Finally, Section V brings the paper to a conclusion while highlighting future direction.

II. BACKGROUND AND RELATED WORK

A. SDN BACKGROUND

The SDN has many advantages such as centralised monitoring that helps in reducing manual communication with the hardware for enhanced network efficiency. Additionally, separating the control plane and data plane leads to simpler hardware and increases the chances of having more expertise among hardware vendors, as the devices do not rely on commercial software [15]. The infrastructure, control (SDN controller), and application layers are the three basic components of SDN architecture. Numerous networking devices, such as switches and routers, make up the infrastructure layer. The control layer is the core of the SDN model where the centralised SDN controller software is hosted. The application layer embodies the implementation of typical workings of the network or functions [16]. Fig. 1 presents the SDN architecture where the OpenFlow is considered the first SDN standards. Essentially, it presents the connection protocol in SDN environments. It is obvious that the OpenFlow separates the infrastructure layer from the control layer. This is highly beneficial, where developers can modify and develop the application layer to meet their needs. Thus, the application layer can appropriately be adapted to the changing business requirements [17].

In addition, OpenFlow is a protocol used to facilitate the connection between network switches and server with respect to received and sent packets. It allows sharing of the same physical infrastructure with numerous logical networks. Besides, the network virtualisation layer includes a collection of controllers for managing a large number of switches. In this case, one switch can belong to numerous virtual networks, controllable through one or more collection of controllers.

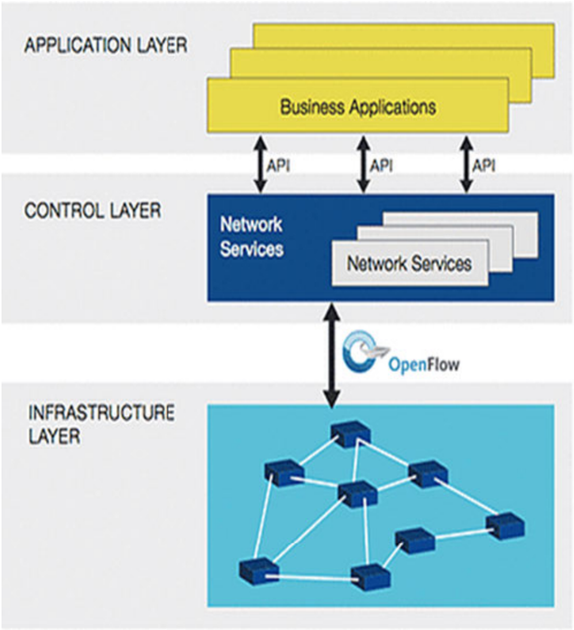


FIGURE 1. The SDN architecture [20].

TABLE 1. Summary of related work.

Year	Algorithm	Dataset	Application	Accuracy	Ref.
2020	Very Fast Decision Tree	MAWI UNI1	Flow Detection	98.64% 99.78%	[21]
2019	SVM DT Random Forest K-Nearest Neighbours	Kaggle	Traffic Classification	96.37% 95.76% 94.92% 71.47%	[22]
2019	SVM Naïve Bayes K-Nearest Neighbours	Flow Generation	Security	-	[23]
2019	Efficient Sampling and Classification Approach	Flow Generation	Network optimization performance	90%	[24]
2019	Deep learning	UCI	Security	85%	[25]
2018	SVM	Flow Generation	Traffic Management	-	[26]
2018	SVM	CAIDA DDoS	Security	-	[27]
2018	SVM Naïve Bayes	Flow Generation	Traffic Classification	-	[28]
2018	DT	Flow Generation	Flow Classification	-	[29]
2017	SVM K-Means	-	Traffic Classification	98% 88%	[30]

However, such a design is susceptible to flow conflicts [18]. Nowadays, depending on the destination, data transmission is redirected. While this approach provides an efficient application of narrowest routing protocols, it does not provide fine-grained network traffic control. Nevertheless, there are several suggestions for future internet designs which need network data plane to perform routing and forwarding at the level of single connections or their aggregate (e.g., network services or network virtualisation) [19].

B. ML ALGORITHMS IN SDN

Machine learning (ML) algorithms have opened up several significant opportunities in the implementation of SDN models; particularly in security applications. These algorithms have widely been used to elevate the performance of SDN models. Here, we will look at the most up-to-date models for traffic classification, flow detection and classification, security, and traffic management, all of which used different machine learning algorithms. Table 1 includes a summary of similar machine learning algorithms used in SDN models. In [21], a flow-aware elephant flow detection is implemented in SDN. To effectively accomplish a reliable elephant flow identification, the proposed approach utilised two classification models simultaneously; One on the SDN switches (i.e., switch-side classifier) and the second on the controller (i.e., controller-side classifier). In addition, this strategy facilitates the exchange of elephant flow identification activities between the controller and switches. Thus, several mouse flows are screened in the switches, thereby eliminating the need for the controller to carry out vast quantities of classification demands and signalling notifications. Experimental results show that, in terms of running time, precision, F-measure, and recall, the proposed methodology outperforms contemporary approaches.

The study conducted in [22] was intended to demonstrate in principle the integration of machine learning with SDN applications for detecting network traffic. It was demonstrated that traffic classification using machine learning algorithms improves performance in the context of SDN. This was largely achievable due to the potential of this structure to gather knowledge. This approach proved highly successful and showed that these high-performing, intelligent-based communication principles can boost or even replace traditional network controls in the near future. In [23], the influence of various OpenFlow time windows on the output prediction of various classification algorithms was discussed. On OpenFlow flow datasets generated in both virtual and physical SDN environments, a total of 150 prototypes were built and tested. The results of the analysis showed that the OpenFlow traffic time interval chosen has a major impact on detection performance with wider time windows resulting in lower detector output. Moreover, by adding correct time-windows to OpenFlow traffic, the authors showed that good precision in detecting unidentified threats can be achieved.

Furthermore, [24] suggested an intelligent solution to screening and classification (ESCA). The authors proposed a modern differentiated scheduling method that independently and progressively establishes routes for elephant and mouse flows. ESCA significantly reduces processing overhead and efficiently classifies specimens using a new supervised classification algorithm of data flow similarities by measuring the delivery time of elephant flows and filtering out duplicate specimens. With a focus on low-cost ESCA, a DiffSch feature-aware flow schedules solution that distinguishes between elephant and mouse flow schedules was proposed. According to the general theory, ESCA outperforms related frameworks. Comprehensive experimental results demonstrate the capability of ESCA to produce accurate identification with far less collected samples with a short detector period, and that certain DiffSch schedule method model outperformed related proposals significantly. In [25], the authors proposed CyberPulse which is a new powerful preventive measurement method that underpins a classifier based on machine learning to mitigate LFA in SDN. By classifying network traffic using deep learning methods, CyberPulse which is incorporated in the Floodlight controller as an enhanced subsystem as opposed to existing techniques on produced practical networks using Mininet conducts network monitoring with impressive precision, false positive rate, and efficiency when evaluated. According to the results, CyberPulse could identify suspicious flows with high accuracy and thus, easily mitigate them. In addition, [26] discussed issues related to flow management caused by network connectivity. A supervised learning prototype was proposed to reduce the SDN controller's reaction time for large complex architectures; thereby allowing the controller to forecast node mobility and connection failure risk. An alternate path preference structure is alternatively introduced to ensure efficient traffic balancing while minimising the workload of the control plane. In the commonly utilised network simulator-ns-3, the result of the proposed algorithm, SDN based Wireless Mesh Network (SD-WMN) model, was verified. Experimental findings demonstrate that the designed SD-WMN model with link-failure proactive traffic management obtained data transmission improvement. The author of [27] developed a system for detecting and deploying DDoS threats in SDN-focused virtual networks. The suggested framework includes not only the control function dependent on the OpenFlow interface statement (i.e., PACKET IN statement) for a non-timely reply, but also a multi-dimensional information-based flow feature extraction method. In addition, creating an efficient nationwide network flow table component behaviour focused on the OpenFlow table function and the flow table entrance stability feature. Evaluation of all feedbacks to the flow table was done by SVM which efficiently decreases the time for initial attack detection and classification identification by evaluating the test outcomes with a smaller false alarm rate. References [28] proposed SDN-Home Gateway (SDN-HGW), which expands the regulation for improved end-to-end network security of the network connection (i.e., a housing automation system). Through the classification of data flows in a smarter housing system, the suggested SDN-HGW gains decentralised device knowledge. There are many current traffic identification approaches for coded data packets. Encoded data classification model known as DataNets is built on several deep learning models to solve these problems using an open data library of around 200,000 sets used for deep learning. The experimental findings indicate that the built DataNets can be used in upcoming smarter housing networking to allow distributed framework SDN-HGW.

Although, machine learning traffic flow classification methods are commonly used, SDN rules are still detected based on the flow categories produced; thus, [29] presented a platform that exacerbates this difficulty. Supervised learning methods was used for various forms of traffic depending on pre modelling techniques. Unsupervised learning was also used to cluster varying traffic flows before, finally, a flow grouping classifier that defines flows that are normally observed together in an identical time period upon identifying the flows. For classification problems, C4.5 decision tree classifiers with functions for each flow, like cross arrival time, packet size, packet number, and flow tuple, are used. In [30], two machine learning algorithms, SVM and K-means, were evaluated for network traffic identification. It was stated that it is possible to obtain an average precision of around 95 percent. Meanwhile, through design adjustment and data preprocessing, the efficiency of the machine could be further improved. Thus, configuration and feature choice of the SVM model was carried out for traffic classification. Findings demonstrate that the radial base kernel function based SVM model resulted in the highest precision and are most effective in numerical terms.

In sum, the findings from all the studies that used machine learning algorithms in the SDN models can be summarised as follow:

- Supervised learning methods are widely employed. Although, KNN, SVM, DTs and Bayesian approaches have higher research interest and are featured in most solutions, there is very little literature on logistic regression usage in reported SDN models.
- Most of the supervised learning algorithms obtain a relatively high average accuracy of over 90% in detection performance across the evaluation metrics.
- In SDN applications, most studies have used SVM and DT algorithms.
- No machine learning implementation exist for the detection and classification of flow conflicts.
- There are various types of datasets been used. While some studies used datasets from internet sources such as Kaggle, others used flow generation method to create the dataset for machine learning algorithms.
- There are no studies showing the key features of flow entries in SDN (e.g., priority and action features) for all forms of dataset used in the current machine learning solutions.
- Precision, recall, and f1-measure are the most commonly used evaluation metrics for validating ML algorithms in most studies. On the other hand, the use of accuracy and execution time remain negligible.

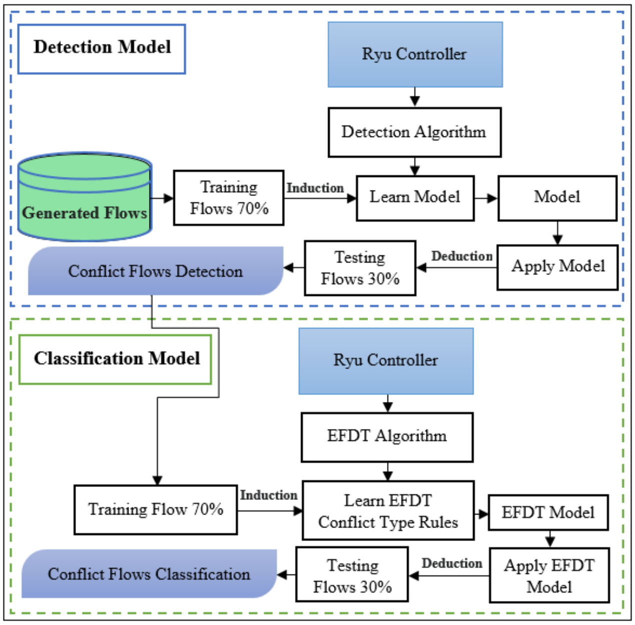


FIGURE 2. **The proposed model for the detection and classification of conflict flows.**

III. MATERIALS AND METHODS

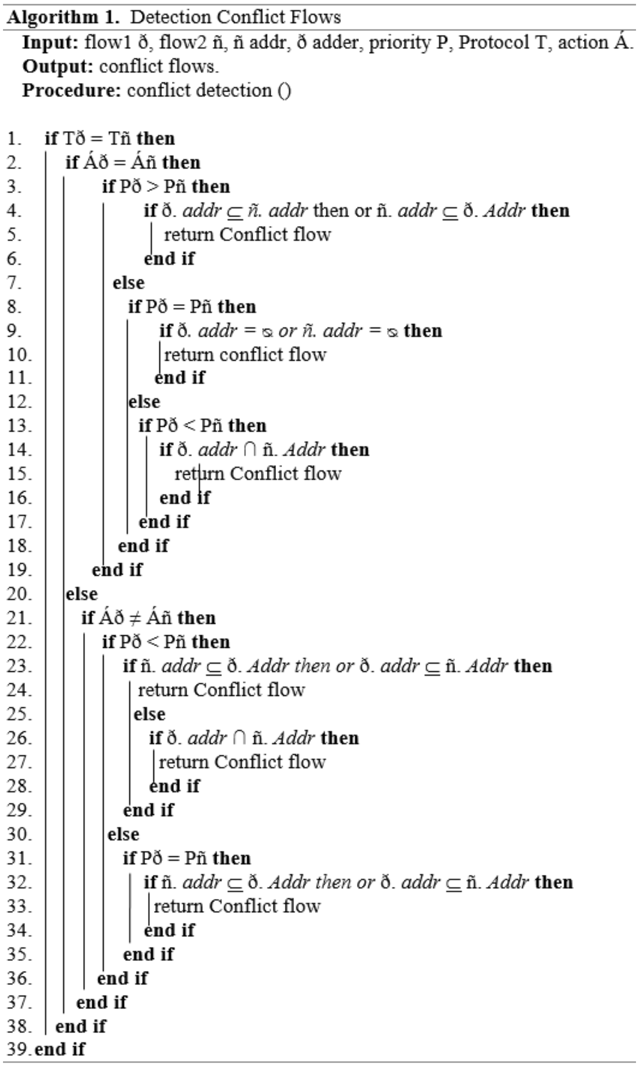
The proposed model in this study has two main phases; detection and classification phases. Fig. 2 shows the proposed model for the detection and classification of conflict flows. The first phase is the detection between conflict flows and normal flows.

In this phase, the generated flow is checked by the algorithms implemented in the controller plane to observe the behaviour of flows. There are features in the flows which are significant in differentiating between normal and conflict flows such as Mac address, IP address and action. Accordingly, the outcome of these feature checking algorithms identifies whether the flows are normal or conflicting. The normal flows pass directly to "OpenFlow" while the conflict flows are forwarded to the next phase for onward classification into which types of conflict occurs in conflict flows.

There are four algorithms proposed for the detection of the conflict flows in "OpenFlow". These algorithms are DT, SVM, EFDT and Hybrid (DT-SVM). The EFDT and hybrid algorithms were developed and implemented from the DT and SVM to improve their performance in terms of accuracy and running time.

The DT and SVM algorithms have been selected because they have shown high performance in previous research in different applications of SDN [31]–[34]. Fig. 3 shows the pseudo code for algorithms used in the detection of conflict flows. Furthermore, the steps involved in the detection phase can be summarised as follow:

1) Implement and run algorithms.
2) The algorithms check the features of flows.
3) The algorithms identify the normal flows and conflict flows.
4) Normal flows are passed as normal to OpenFlow.



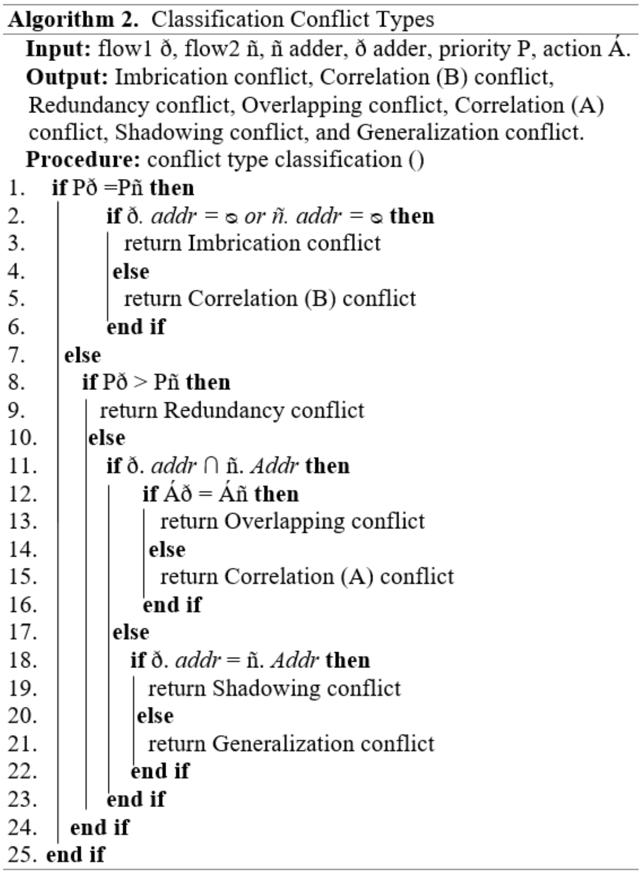
```
Algorithm 1. Detection Conflict Flows
  Input: flow1 ð, flow2 ñ, ñ addr, ð adder, priority P, Protocol T, action Á.
  Output: conflict flows.
  Procedure: conflict detection ()

if Tð = Tñ then
  if Áð = Áñ then
      if Pð > Pñ then
          if ð. addr ⊆ ñ. addr then or ñ. addr ⊆ ð. Addr then
            return Conflict flow
          end if
        else
          if Pð = Pñ then
             if ð. addr = ∅ or ñ. addr = ∅ then
               return conflict flow
             end if
          else
           if Pð < Pñ then
             if ð. addr ∩ ñ. Addr then
               return Conflict flow
             end if
           end if
        end if
     end if
  else
    if Áð ≠ Áñ then
      if Pð < Pñ then
        if ñ. addr ⊆ ð. Addr then or ð. addr ⊆ ñ. Addr then
          return Conflict flow
         else
          if ð. addr ∩ ñ. Addr then
            return Conflict flow
          end if
        end if
      else
        if Pð = Pñ then
          if ñ. addr ⊆ ð. Addr then or ð. addr ⊆ ñ. Addr then
            return Conflict flow
          end if
        end if
      end if
    end if
  end if
end if
```

FIGURE 3. **The pseudo code for conflict flows detection.**

5) The conflict flows are passed to the classification algorithm.

The second phase of the proposed model is the classification of conflict flows. In this phase, the conflict flows identified in the detection phase are checked by an algorithm implemented in the controller plane to determine the behaviour of flows. The three features of conflict flows are priority, IP address, and action. Upon completion of the checking process, the conflict types are classified into seven types which are redundancy, shadowing, overlapping, correlation A, correlation B, generalisation, and imbrication. Fig. 4 shows the pseudo code for the EFDT algorithm in classifying conflict flows. Moreover, the steps involved in the classification phase can be summarised as follows:

1) Implement and run the EFDT classifier algorithm.
2) The algorithm begins detection of flows.
3) The algorithm checks the priority and IP address features of the flows.



```
Algorithm 2. Classification Conflict Types
Input: flow1 ð, flow2 ñ, ñ adder, ð adder, priority P, action Á.
Output: Imbrication conflict, Correlation (B) conflict,
Redundancy conflict, Overlapping conflict, Correlation (A)
conflict, Shadowing conflict, and Generalization conflict.
Procedure: conflict type classification ()
if Pð =Pñ then
    if ð. addr = ∅ or ñ. addr = ∅ then
        return Imbrication conflict
    else
        return Correlation (B) conflict
    end if
else
    if Pð > Pñ then
        return Redundancy conflict
    else
        if ð. addr ∩ ñ. Addr then
            if Áð = Áñ then
                return Overlapping conflict
            else
                return Correlation (A) conflict
            end if
        else
            if ð. addr = ñ. Addr then
                return Shadowing conflict
            else
                return Generalization conflict
            end if
        end if
    end if
end if
```

FIGURE 4. The pseudo code for classification conflict flows detection.

4) The algorithm classifies the conflict types according to the features in step 3.

A. DECISION TREE (DT) ALGORITHM

Decision Tree is a machine learning algorithm that is implemented to handle regression and classification problems. However, it is more often utilised for classification tasks. It is a tree-structured algorithm where the characteristics of a database are described through internal nodes and branches that represent the decision rules while the outcome is indicated by each leaf node. Decision nodes are used to make decisions and thus, have several branches. The outcome of these decisions are Leaf nodes with no additional branches. Fig. 5 shows the diagram that illustrates the general structure of the DT algorithm. It is also possible to describe decision trees as a mix of mathematical and analytical methods to help identify, categorise and generalise a given data set. Data comes in the form of records as shown in the following equation:

$$(x, Y) = (x1, x2, x3 \ldots . xn, Y) \quad (1)$$

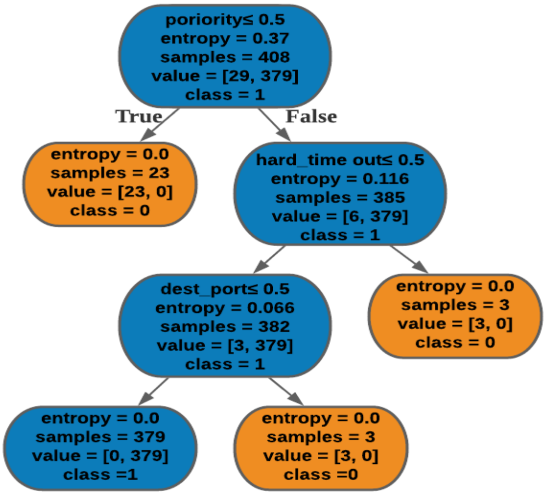


FIGURE 5. The diagram of DT algorithm.

The conditional factor Y is the reference parameter that learning attempts to describe or categorise. The vector x is made up of the characteristics $x1, x2, x3$ etc. that are used for learning. Additionally, the implementation steps of DT algorithm can be summarised as follow:

- Implement decision tree components in the controller plane.
- Setup the learner function.
- Prepare and import all generated flows from OpenFlow switch for all flow sizes.
- Train the algorithm with 70% of generated flows.
- Test the performance of the algorithm by predicting the response for remaining 30% of generated flows.
- Evaluate the confusion matrix for the DT algorithm and calculate running time.

B. SUPPORT VECTOR MACHINE (SVM) ALGORITHM

The Support Vector Machine (SVM) is a binary supervised classifier employed in Machine Learning. The aim of the SVM algorithm is to build near perfect lines or decision boundaries to divide an n-dimensional area into categories so that specific data points can be easily placed in the appropriate category in the future. A hyper-plane is a term used to describe the best decision boundary. SVM selects unique points/vectors that aid in the construction of the hyperplane. Help vectors are a term used to describe these extreme situations. In the classification method, two distinct classes use a decision boundary or hyper-plane, as shown in Fig. 6.

A learning database of n points is as in the following formula

$$(\mathrm{x}_1, \mathrm{y}_1) \ldots . (\mathrm{x}_\mathrm{n}, \mathrm{y}_\mathrm{n}) \quad (2)$$

where y_n can either be 1 or -1; showing which category the x_i belongs to. Every x_i is a valid ***p* − *dimensional*** vector. The segment to which x_i corresponds as indicated by either 1 or -1 is y_n. Each hyperplane can be defined as a collection of nodes that satisfy x_i.

$$w^T x - b = 0 \quad (3)$$

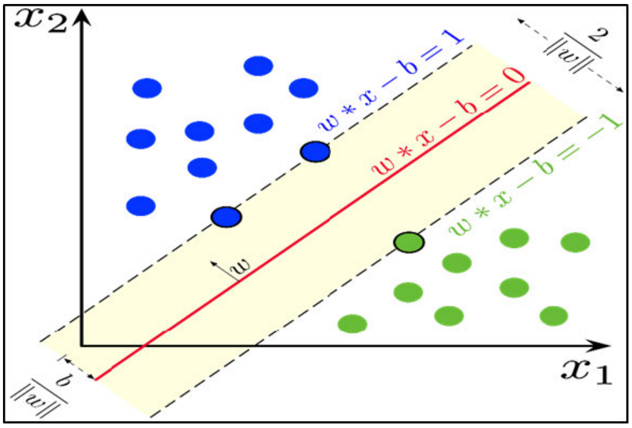


FIGURE 6. The two distinct groups for SVM algorithm.

The standard vector to the hyperplane is w. This is also identical to the normal state of Hesse except that w is not actually a unit vector. Component b/(||w||) specifies the offset of the hyperplane from its source across the standard vector w. In Equation 4, every point on or above that boundary belongs to a single class, marked 1. While in Equation 5, all points on and below that boundary marked with −1 is of the other class.

$$w^T x - b = 1 \tag{4}$$

$$w^T x - b = -1 \tag{5}$$

The range between the two hyper-planes is geometrically 2/(||w||). It is important to reduce the ||w|| in order to increase the gap between the planes. The distance from a point to a plane is calculated using the distance. To prevent sets of data from falling into the margin, it also imposes the following constraints as in Equation 6 or 7 for each i.

$$w^T x_i - b \geq 1, \quad y_i = 1 \tag{6}$$

Or

$$w^T x_i - b < 1, \quad y_i = -1 \tag{7}$$

Based on these constraints, each information pointed should on the right location of the line. This can be rewritten as the following equation.:

$$y_i(w^T x_i - b) \geq 1, \quad \text{for all } 1 \leq i \leq n \tag{8}$$

Furthermore, the implementation steps of the SVM algorithm can be summarised as follows:

- Implement support vector components in the Ryu controller.
- Setup the learner of the linear module.
- Apply hard margin function.
- Prepare and import all generated flows from OpenFlow switch for all flow sizes.
- Train the SVM classifier with 70% of generated flows.
- Test the performance of the classifier by predicting the response for the remaining 30% of generated flows.
- Evaluate the confusion matrix for the SVM algorithm and calculate running time.

C. EXTREMELY FAST DECISION TREE (EFDT)

The Extremely Fast Decision Tree (EFDT) is a novel learning algorithm which when implemented with the Hoeffding Anytime Tree SEA Generator, is systematically more effective than the conventional decision tree algorithm. On several traditional benchmark tasks, the EFDT outperforms the Hoeffding Tree implementation of Very Fast Decision Tree (VFDT) in terms of prequential accuracy. Domingos and Hulten implemented Hoeffding Tree; one of the first algorithms for progressively constructing a decision tree in their highly lauded research [35]. Hoeffding Tree checks whether the difference between the average information improvements of the highest two parameters is going to provide a great meaning in almost any given potential break.

Hoeffding Bound: If n is independent random variables $r_1 \ldots r_n$, with a wide variety R and mean $\check{r}$, the Hoeffding bound declares in conjunction with probability $1 - \delta$ the real mean is at the very minimum $\check{r} - \epsilon$ [36].

$$\epsilon = \frac{\sqrt{R^2 ln(\frac{1}{\delta})}}{2n} \tag{9}$$

The Hoeffding Tree uses this deterministic guarantee to determine if the calculated variation of information changes is between the Xa and Xb attributes with the maximum data gains, respective, around each node. Thus, $\Delta \breve{G}$ $(\boldsymbol{X_a})$ – $\Delta \breve{G}$ $(\boldsymbol{X_b})$, is positive and non-zero. Unless, for the tolerance stated, δ, it has $\Delta \breve{\mathbf{G}} > \boldsymbol{\epsilon}$, then it confidently declares that $\boldsymbol{X_a}$ is the more advantageous division. It is worth noting that it aims to determine the best selection segment. The probabilities are monitored in the manner described before that $\boldsymbol{X_a}$ is superior to $\boldsymbol{X_b}$. However, the probability that $\boldsymbol{X_a}$ is superior to any other $\boldsymbol{X_c}$ feature is not regulated. If the selection of attributes increases, it becomes more likely that every other category will be better. In such situation, there is no recourse to modify the tree. Furthermore, the implementation steps of the SVM algorithm can be summarised as follows:

- Setup the SEA Generator into DT components.
- Implement the Hoeffding Tree in the classifier.
- Setup and modify the Hoeffding Tree estimator to check action and IP address rules for the generated flows.
- Setup new variables to control the loop for checking action and IP address rules.
- Prepare and import all generated flows from OpenFlow switch for all flow sizes.
- Train the EFDT algorithm with 70% of generated flows.
- Test the performance of the algorithm by predicting the response for the remaining 30% of generated flows.
- Evaluate the confusion matrix for the EFDT algorithm and calculate running time.

D. HYBRID (DT-SVM) ALGORITHM

The learner of two algorithms was designed and implemented as one learner to enhance the precision and execution time. The hybrid algorithm has been purposefully implemented from two algorithms, decision tree classifier

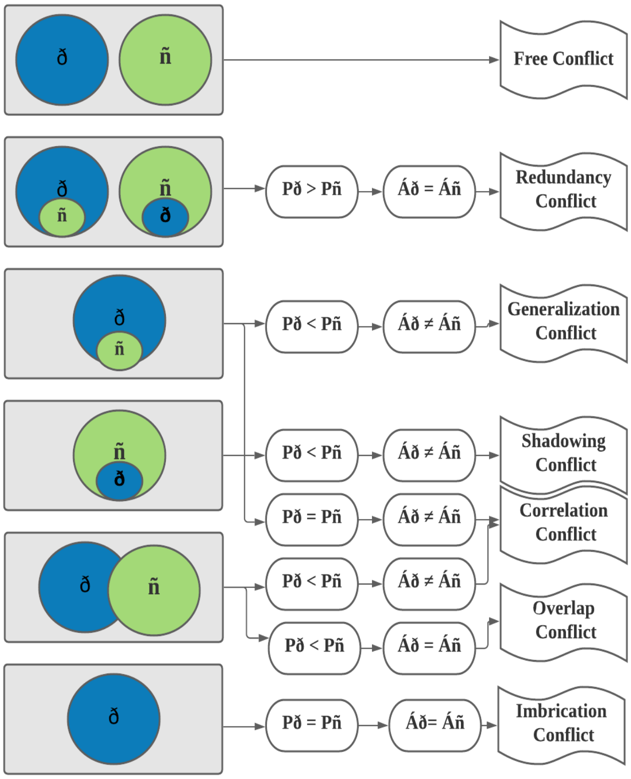


FIGURE 7. The conflict flow types.

and super vector classifier, to enhance the performance of the two algorithms. One-vs-the-rest (OvR), also known as one-vs-all, multiclass strategy is utilised. This consists of fitting one classifier per class. Notable merits of this method are its interpretability as well as fast computation (only n-class classifiers are required). While each class is identified by one classifier, it is imperative to obtain information of the class by examining the related classifier. This is the most widely used multiclass classification technique and is a rational choice by default. Furthermore, the implementation steps for the hybrid DT-SVM algorithm can be summarised as follows:

- Implement the DT classifier object together with the SVM classifier.
- Implement the OvR classifier.
- Setup and modify the OvR classifier for DT and SVM classifiers for action and IP address rules.
- Integrate DT and SVM classifiers.
- Setup and implement voting classifier to combine the prediction of DT and SVM classifiers.
- Prepare and import all generated flows from OpenFlow switch for all flow sizes.
- Train the hybrid DT-SVM algorithm with 70% of generated flows.
- Test the performance of the algorithm by predicting the response for the remaining 30% of generated flows.
- Evaluate the confusion matrix for the hybrid DT-SVM algorithm and calculate running time.

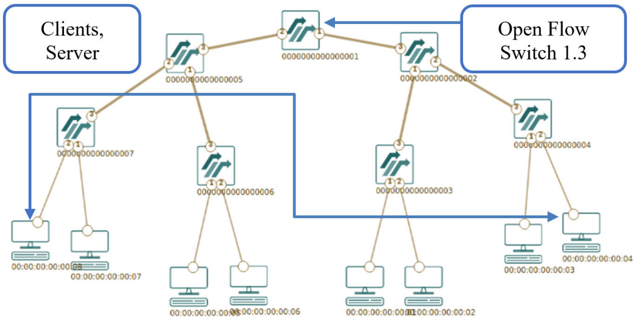


FIGURE 8. Fat tree topology.

IV. EXPERIMENT RESULTS AND DISCUSSION

We have considered SDN dataset from our past studies in this analysis [37] for normal and conflict flows. There are seven types of conflict flows namely redundancy, shadowing, overlapping, correlation A, correlation B, generalisation, and imbrication. Fig. 7 shows the conflict flow types used in this study.

The pre-processing model is introduced and implemented to prepare flows collected from the OpenFlow switch for the detection process. The flows were extracted to present the important features (action, protocol, mac address, and IP address) which are chosen for training the algorithms. The type of conflict can be specified and classified according to priority, action, protocol and IP source address of the flow rule. Flows will be deemed conflicting flow entries as per the flow rule in the open flow switch. SDN can be influenced by conflict in various situations which ultimately affects the efficiency and optimisation of the network in the form of redundancy, overlap and correlation conflict. Moreover, it can also affect the security of the entire network; leading to shadowing generalisation and imbrication conflict [38].

Furthermore, there are two topologies used in this study, namely; Fat Tree Topology and Simple Tree Topology. The Ryu controller is used in this experiment to create a link to an OpenFlow switch version 1.3 to enable both topologies to analyze data. These two topologies were created in mininet and then connected to the Ryu controller to automatically generate the traffic. Fig. 8 and Fig. 9 show the architecture of Fat Tree and Simple Tree topologies respectively. Besides, the Fat Tree topology contains 7 switches and 8 hosts while the Simple Tree topology contains 3 switches and 4 hosts. The Ryu controller is associated to all switches and hosts in these topologies. Topo.py, a Python application that connects switches and hosts in these topologies, is programmed and deployed over a python programming language. Regarding the production of flows, traffic generation is performed to produce flows in the range of 1000-100000 flows (i.e., it starts at 1000 flows and finished at 100000 flows) at intervals of 10000 flows increment.

Every host starts with 10 iperf servers, each of which listens to different ports such as 8089, 8082, and 8081. A simple switch is needed in the flow entry production steps. The L4 Match application was created chosen as the basis framework.

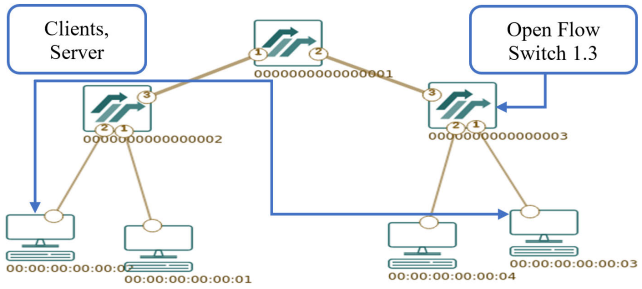


FIGURE 9. Simple tree topology.

The src/dst ip, src/dst port, and protocols have been utilized to build different flows. The controller receives each packet and then creates a new flow in the switch. Generally, after the topologies were created by running the *Topo* app, the number of flows are selected after which the Ryu manager app starts running to generate normal flows. After the given number of flows are generated, the conflicts rules are implemented in the Ryu controller by running the conflicts flow app. When all generation of normal and conflict flows were completed, the flowstat app is performed to collect and save all flows generated in a CSV file. Fig. 10 shows the flowchart used to produce and generate the flows.

Both tests were carried out on a PC running Ubuntu 18.04, with an Intel Core-i5 CPU and 12 GB of RAM within Python 2.7 programming language environment. We utilized a variety of assessment measures that include accuracy, precision, f1-score, recall, and execution time to evaluate the performance of the proposed algorithms during the identification and classification of conflict flows in terms of efficiency and effectiveness. These evaluation measurements are computed as shown in Equations (10-14).

$$Accuracy = \frac{TP + TN}{(TP + TN + FN + FP)} \tag{10}$$

$$Precision = \frac{TP}{TP + FP} \tag{11}$$

$$F1 - score = 2 \times \frac{(precision \times recall)}{(precision + recall)} \tag{12}$$

$$Recall = \frac{TP}{TP + FN} \tag{13}$$

$$Execution\ Time\ (T) = T_{start} T_{finish} \tag{14}$$

- The number of conflicts flows correctly classified are referred to as true positives (TPs).
- The number of correctly categorised normal flows are known as true negatives (TNs).
- The number of natural flows that are incorrectly labelled as conflict flows are known as false positives (FP).
- The number of conflict flow instances that are incorrectly categorised as normal flows are known as false negatives (FN).

The output of the suggested implementations reveals a variety of experimental findings. When the number of flows was 1000, the DT, SVM, and hybrid DT-SVM algorithms achieved the highest detection results with respect to accuracy precision, f1-score, and recall. The highest detection accuracies for DT, SVM, and hybrid DT-SVM algorithms are respectively 99.27%, 98.53%, and 99.27%. In addition, the highest detection results for the EFDT algorithm were achieved when the number of flows was 100000; yielding a maximum detection accuracy of 99.49%. It is worthy of note that the EFDT algorithm yielded the highest possible value of 100% for precision, f1-score, and recall respectively. Furthermore, the lowest execution time for all algorithms (i.e., DT, SVM, hybrid DT-SVM, and EFDT) was 0.00021 seconds at 20000 flows. It is obvious now that the EFDT algorithm produced the best results compared to the results of DT, SVM, and hybrid DT-SVM algorithms. However, when the number of flows was 10000, the minimum detection accuracies for DT, SVM, and hybrid DT-SVM algorithms

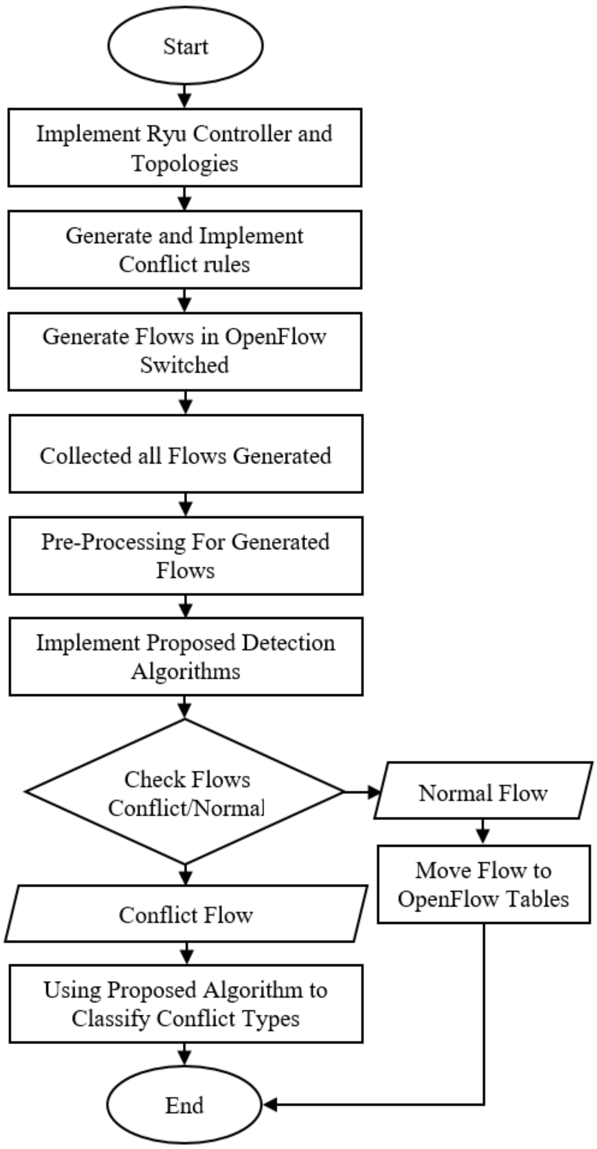

Start
Implement Ryu Controller and Topologies
Generate and Implement Conflict rules
Generate Flows in OpenFlow Switched
Collected all Flows Generated
Pre-Processing For Generated Flows
Implement Proposed Detection Algorithms
Check Flows Conflict/Normal
Normal Flow
Move Flow to OpenFlow Tables
Conflict Flow
Using Proposed Algorithm to Classify Conflict Types
End

FIGURE 10. Flowchart of flow generation.

TABLE 2. Detection results by using all algorithms.

DT Algorithm					
Dataset	Accuracy	Precision	F1-score	Recall	Time (sec)
1000	**99.27%**	**98%**	**98%**	**97%**	**7.64E-06**
10000	72.60%	57%	61%	67%	7.63E-05
20000	81.74%	60%	63%	67%	0.00021
30000	81.37%	60%	63%	67%	0.01055
40000	82.08%	45%	47%	50%	0.01082
50000	81.26%	45%	47%	50%	0.01116
60000	81.08%	60%	63%	67%	0.01157
70000	80.91%	45%	47%	50%	0.01205
80000	81.43%	60%	63%	67%	0.01259
90000	81.03%	45%	47%	50%	0.0132
100000	80.48%	60%	63%	67%	0.01387
SVM Algorithm					
Dataset	Accuracy	Precision	F1-score	Recall	Time (sec)
1000	**98.53%**	**99%**	**95%**	**91%**	**7.73E-06**
10000	64.60%	64%	64%	65%	7.64E-05
20000	77.08%	67%	66%	67%	0.00021
30000	70.97%	65%	65%	65%	0.00042
40000	82.02%	45%	47%	50%	0.00069
50000	81.26%	45%	47%	50%	0.00103
60000	81.08%	60%	63%	67%	0.00144
70000	80.91%	45%	47%	50%	0.00191
80000	81.43%	60%	63%	67%	0.00245
90000	81.03%	45%	47%	50%	0.00306
100000	80.48%	60%	63%	67%	0.00374
EFDT Algorithm					
Dataset	Accuracy	Precision	F1-score	Recall	Time (sec)
1000	94.01%	92%	96%	**100%**	**7.44E-06**
10000	97.95%	98%	98%	99%	7.58E-05
20000	98.85%	99%	99%	99%	0.00021
30000	99.10%	99%	99%	99%	0.00041
40000	99.21%	99%	99%	**100%**	0.00069
50000	99.29%	99%	99%	**100%**	0.00102
60000	99.36%	**100%**	**100%**	**100%**	0.00143
70000	99.38%	**100%**	**100%**	99%	0.0019
80000	99.43%	**100%**	**100%**	**100%**	0.00248
90000	99.46%	**100%**	**100%**	**100%**	0.00308
100000	**99.49%**	**100%**	**100%**	**100%**	0.00377
Hybrid DT-SVM Algorithm					
Dataset	Accuracy	Precision	F1-score	Recall	Time (sec)
1000	**99.27%**	**99%**	**95%**	**91%**	**7.59E-06**
10000	72.60%	57%	61%	67%	7.73E-05
20000	81.74%	60%	63%	67%	0.00021
30000	81.37%	60%	63%	67%	0.00043
40000	82.08%	45%	47%	50%	0.0007
50000	81.26%	45%	47%	50%	0.00104
60000	81.08%	60%	63%	67%	0.00145
70000	80.91%	45%	47%	50%	0.00192
80000	81.43%	60%	63%	67%	0.00248
90000	81.03%	45%	47%	50%	0.00309
100000	80.48%	60%	63%	67%	0.00377

TABLE 3. Classification results by using EFDT algorithm.

Dataset	Accuracy	Precision	F1-score	Recall	Time (sec)
1000	90.16%	**97.61%**	91.56%	86.47%	**2.96E-02**
10000	**95.73%**	95.85%	**96.64%**	97.61%	0.3248
20000	93.58%	91.76%	95.35%	99.26%	1.0836
30000	93.75%	91.54%	95.58%	**100%**	2.31071
40000	93.31%	91.33%	95.24%	99.55%	14.2404
50000	94.08%	92.10%	95.79%	99.81%	16.4857
60000	95.05%	93.47%	96.50%	99.78%	19.1472
70000	94.68%	92.75%	96.23%	**100%**	22.3985
80000	94.48%	92.54%	96.12%	**100%**	26.0132
90000	94.32%	92.27%	95.98%	**100%**	30.1229
100000	93.99%	92.49%	95.65%	99.04%	34.7073

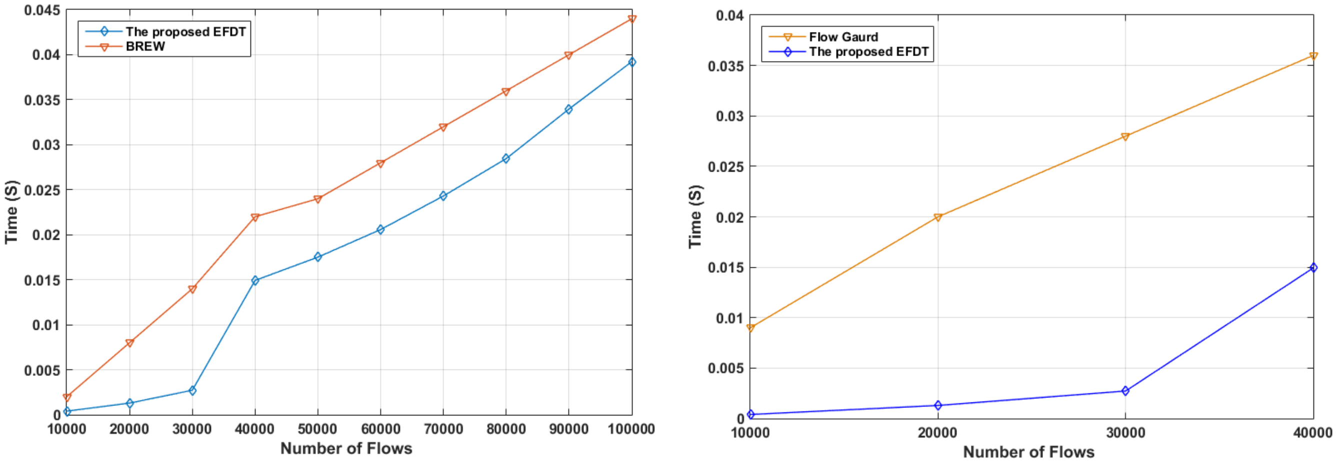


FIGURE 11. Comparison of execution time between EFDT and other methods.

were 72.60%, 64.60%, and 72.60%, respectively while the minimum detection accuracy for the EFDT algorithm was 94.01% at 1000 flows. Moreover, the longest execution time was more than 7 seconds for all algorithms when the number of flows were 1000 and 10000. Table 2 shows the detection results for DT, SVM, hybrid DT-SVM, and EFDT algorithms, respectively.

The improved performance resulted from the constituent processes of the implemented EFDT algorithms. This include the Hoeffding Bound that was used to improve detection accuracy, the implemented modifications, and the setup of Hoeffding Tree estimator to check the action and IP address of flows to speedup detection time with improved detection accuracy.

According to the detection results, the EFDT algorithm has been selected for the classification phase to identify the types of conflict flows. The EFDT algorithm was chosen because it has the best performance in detecting conflict flows. During the classification process, flows within a range of 1000 to 100000 were chosen, with a 10000-flow multiplier. The performance of the proposed EFDT algorithm in classifying types of conflict flows achieved the highest results based on a different number of flows. The highest accuracy and f1-score achieved by the EFDT algorithm are respectively 95.73% and 96.64% with 10000 flows while the best precision achieved was 97.61% with 1000 flows. However, the highest recall achieved was 100% when the flows were 30000, 70000, 80000, and 90000 respectively. Also, the lowest execution time taken for the classification using EFDT algorithm was 0.3248 second. Additionally, the minimum classification accuracy was 90.16% when the flows were 1000. Table 3 shows the classification results using the proposed EFDT algorithm.

The classification algorithm shows the best result in all conflict flows sizes across all evaluated metrics. The improved performance resulted from the constituent processes of the implemented EFDT algorithms. This include the Hoeffding Bound that was used to improve detection accuracy, the implemented modifications and the setup of Hoeffding Tree estimator to check the action and IP address of flows to speedup detection time with improved detection accuracy. A comparative analysis with two benchmarks shows the validation of the proposed EFDT algorithm for the detection and classification across all flow sizes data.

Furthermore, the suggested EFDT algorithm was compared with other methods in [39] and [40] in terms of the time taken for detection and classification of conflict flows. References [39] was implemented in the security policy analysis using the Brew module. The number of flows in this work is selected between 10000 to 100000 flows. Fig. 11 (a) shows the comparison of execution time between the proposed EFDT algorithm and the Brew module. References [40] has also presented a comprehensive framework called Flow Guard in the OpenFlow networks in which the number of flows selected ranged between 10000 to 40000 flows. The proposed EFDT algorithm and the Flow Guard method are also compared in terms of execution time in Fig. 11 (b). The performance of the frameworks presented in both studies were also evaluated on the detection and classification of conflict flows. The proposed EFDT algorithm outperformed its comparative methods in terms of time taken for the detection and classification of conflict flows.

V. CONCLUSION

This paper presents several machine learning algorithms for detecting and classifying conflict flows in the SDN model. The types of conflict are detected and classified based on the flow rules' priority, action, protocol, and IP source address. The four algorithms that were utilized in this research are Decision Tree (DT), Support Vector Machine (SVM), Extremely Fast Decision Tree (EFDT), and the Hybrid (DT-SVM); where the proposed EFDT and DT-SVM algorithms were respectively developed based on DT and SVM algorithms to enhance their performance with respect to efficiency and effectiveness. Besides, there were two network topologies designed, namely, Fat Tree Topology and Simple Tree Topology. These network topologies were created using the Mininet simulator and connected to the Ryu controller. The number of flows selected ranged from 1000 flows to 100000 flows with an increment step of 10000 flows for the dataset. The performance of the proposed algorithms was evaluated using evaluation metrics that include accuracy, precision, f1-score, recall, and execution time. Experiment results show that the proposed EFDT algorithm achieved produced the best results compared to DT, SVM, and DT-SVM algorithms with a detection accuracy of 99.49%. In the case of classification between conflict flow types, the proposed EFDT achieved 95.73% accuracy.

The proposed algorithm has been shown to have the capability of achieving promising results in the detection and classification of conflict flows in SDN. To the best of our knowledge, this work is the first attempt at using machine learning algorithms to detect and classify conflict flows. Future works will focus on examining other machine learning algorithms for detecting and classifying conflict flows using the same dataset.

REFERENCES

[1] C.-C. Lo, P.-Y. Wu, and Y.-H. Kuo, "Flow entry conflict detection scheme for software-defined network," in *Proc. Int. Telecommun. Netw. Appl. Conf. (ITNAC)*, Sydney, NSW, Australia, Nov. 2015, pp. 220–225, doi: 10.1109/ATNAC.2015.7366816.

[2] M. H. H. Khairi, S. H. S. Ariffin, N. M. A. Latiff, A. S. Abdullah, and M. K. Hassan, "A review of anomaly detection techniques and distributed denial of service (DDoS) on software defined network (SDN)," *Eng., Technol. Appl. Sci. Res.*, vol. 8, no. 2, pp. 2724–2730, Apr. 2018.

[3] A. Abdelaziz, A. T. Fong, A. Gani, S. Khan, F. Alotaibi, and M. K. Khan, "On software-defined wireless network (SDWN) network virtualization: Challenges and open issues," *Comput. J.*, vol. 60, no. 10, pp. 1510–1519, Oct. 2017.

[4] M. Hamdan, E. Hassan, A. Abdelaziz, A. Elhigazi, B. Mohammed, S. Khan, A. V. Vasilakos, and M. N. Marsono, "A comprehensive survey of load balancing techniques in software-defined network," *J. Netw. Comput. Appl.*, vol. 174, Jan. 2021, Art. no. 102856.

[5] M. A. A. Albadr, S. Tiun, and F. T. Al-Dhief, "Evaluation of machine translation systems and related procedures," *ARPN J. Eng. Appl. Sci.*, vol. 13, no. 12, pp. 3961–3972, Jun. 2018.

[6] M. A. A. Albadra and S. Tiuna, "Extreme learning machine: A review," *Int. J. Appl. Eng. Res.*, vol. 12, no. 14, pp. 4610–4623, 2017.

[7] M. A. Mohammed, S. A. Mostafa, O. I. Obaid, S. R. M. Zeebaree, G. Ghani, A. Mustapha, M. F. M. Fudzee, M. A. Jubair, M. H. Hassan, A. Ismail, D. A. Ibrahim, and F. T. AL-Dhief, "An anti-spam detection model for emails of multi-natural language," *J. Southwest Jiaotong Univ.*, vol. 54, no. 3, pp. 1–14, Jun. 2019.

[8] M. A. A. Albadr, S. Tiun, M. Ayob, F. T. Al-Dhief, K. Omar, and F. A. Hamzah, "Optimised genetic algorithm-extreme learning machine approach for automatic COVID-19 detection," *PLoS ONE*, vol. 15, no. 12, pp. 1–28, Dec. 2020.

[9] O. I. Obaid, M. A. Mohammed, M. K. A. Ghani, A. Mostafa, and F. Taha, "Evaluating the performance of machine learning techniques in the classification of Wisconsin breast cancer," *Int. J. Eng. Technol.*, vol. 7, no. 4.36, pp. 160–166, 2018.

[10] F. T. AL-Dhief, N. M. A. Latiff, N. N. N. A. Malik, N. Sabri, M. M. Baki, M. A. A. Albadr, A. F. Abbas, Y. M. Hussein, and M. A. Mohammed, "Voice pathology detection using machine learning technique," in *Proc. IEEE 5th Int. Symp. Telecommun. Technol. (ISTT)*, Shah Alam, Malaysia, Nov. 2020, pp. 99–104, doi: 10.1109/ISTT50966.2020.9279346.

[11] F. T. Al-Dhief, N. M. A. Latiff, N. N. N. A. Malik, N. S. Salim, M. M. Baki, M. A. A. Albadr, and M. A. Mohammed, "A survey of voice pathology surveillance systems based on Internet of Things and machine learning algorithms," *IEEE Access*, vol. 8, pp. 64514–64533, 2020, doi: 10.1109/ACCESS.2020.2984925.

[12] M. A. Mohammed, K. H. Abdulkareem, S. A. Mostafa, M. K. Ghani, M. S. Maashi, B. Garcia-Zapirain, I. Oleagordia, H. Alhakami, and F. T. Al-Dhief, "Voice pathology detection and classification using convolutional neural network model," *Appl. Sci.*, vol. 10, no. 11, pp. 1–13, May 2020.

[13] M. A. A. Albadr, S. Tiun, F. T. Al-Dhief, and M. A. M. Sammour, "Spoken language identification based on the enhanced self-adjusting extreme learning machine approach," *PLoS ONE*, vol. 13, no. 4, pp. 1–27, Apr. 2018.

[14] M. A. A. Albadr, S. Tiun, M. Ayob, and F. T. AL-Dhief, "Spoken language identification based on optimised genetic algorithm–extreme learning machine approach," *Int. J. Speech Technol.*, vol. 22, no. 3, pp. 711–727, Sep. 2019.

[15] S. Khan, A. Gani, A. W. Abdul Wahab, M. Guizani, and M. K. Khan, "Topology discovery in software defined networks: Threats, taxonomy, and state-of-the-art," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 1, pp. 303–324, 1st Quart., 2017, doi: 10.1109/COMST.2016.2597193.

[16] S. Khan, M. A. Bagiwa, A. W. A. Wahab, A. Gani, and A. Abdelaziz, "Understanding link fabrication attack in software defined network using formal methods," in *Proc. IEEE Int. Conf. Informat., IoT, Enabling Technol. (ICIoT)*, Doha, Qatar, Feb. 2020, pp. 555–562, doi: 10.1109/ICIoT48696.2020.9089520.

[17] Z. Bozakov and V. Sander, "OpenFlow: A perspective for building versatile networks," in *Network-Embedded Management and Applications*, A. Clemm and R. Wolter, Eds. New York, NY, USA, Springer, 2013, pp. 217–245, doi: 10.1007/978-1-4419-6769-5_11.

[18] C. N. Tran and V. Danciu, "On conflict handling in software-defined networks," in *Proc. Int. Conf. Adv. Comput. Appl. (ACOMP)*, Ho Chi Minh City, Vietnam, Nov. 2018, pp. 50–57, doi: 10.1109/ACOMP.2018.00016.

[19] S. Xu, X.-W. Wang, and M. Huang, "Software-defined next-generation satellite networks: Architecture, challenges, and solutions," *IEEE Access*, vol. 6, pp. 4027–4041, 2018, doi: 10.1109/ACCESS.2018.2793237.

[20] A. A. Neghabi, N. Jafari Navimipour, M. Hosseinzadeh, and A. Rezaee, "Load balancing mechanisms in the software defined networks: A systematic and comprehensive review of the literature," *IEEE Access*, vol. 6, pp. 14159–14178, 2018, doi: 10.1109/ACCESS.2018.2805842.

[21] M. Hamdan, B. Mohammed, U. Humayun, A. Abdelaziz, S. Khan, M. A. Ali, M. Imran, and M. N. Marsono, "Flow-aware elephant flow detection for software-defined networks," *IEEE Access*, vol. 8, pp. 72585–72597, 2020, doi: 10.1109/ACCESS.2020.2987977.

[22] M. P. J. Kuranage, K. Piamrat, and S. Hamma, "Network traffic classification using machine learning for software defined networks," in *Proc. Int. Conf. Mach. Learn. Netw.* Cham, Switzerland: Springer, 2020, pp. 28–39, doi: 10.1007/978-3-030-45778-5_3.

[23] S. Khamaiseh, E. Serra, Z. Li, and D. Xu, "Detecting saturation attacks in SDN via machine learning," in *Proc. 4th Int. Conf. Comput., Commun. Secur. (ICCCS)*, Rome, Italy, Oct. 2019, pp. 1–8, doi: 10.1109/CCCS.2019.8888049.

[24] F. Tang, H. Zhang, L. T. Yang, and L. Chen, "Elephant flow detection and differentiated scheduling with efficient sampling and classification," *IEEE Trans. Cloud Comput.*, early access, Feb. 26, 2019, doi: 10.1109/TCC.2019.2901669.

[25] R. U. Rasool, U. Ashraf, K. Ahmed, H. Wang, W. Rafique, and Z. Anwar, "Cyberpulse: A machine learning based link flooding attack mitigation system for software defined networks," *IEEE Access*, vol. 7, pp. 34885–34899, 2019, doi: 10.1109/ACCESS.2019.2904236.

[26] K. Bao, J. D. Matyjas, F. Hu, and S. Kumar, "Intelligent software-defined mesh networks with link-failure adaptive traffic balancing," *IEEE Trans. Cognit. Commun. Netw.*, vol. 4, no. 2, pp. 266–276, Jun. 2018, doi: 10.1109/TCCN.2018.2790974.

[27] Y. Yu, L. Guo, Y. Liu, J. Zheng, and Y. Zong, "An efficient SDN-based DDoS attack detection and rapid response platform in vehicular networks," *IEEE Access*, vol. 6, pp. 44570–44579, 2018, doi: 10.1109/ACCESS.2018.2854567.

[28] P. Wang, F. Ye, X. Chen, and Y. Qian, "DataNet: Deep learning based encrypted network traffic classification in SDN home gateway," *IEEE Access*, vol. 6, pp. 55380–55391, 2018, doi: 10.1109/ACCESS.2018.2872430.

[29] D. Comaneci and C. Dobre, "Securing networks using SDN and machine learning," in *Proc. IEEE Int. Conf. Comput. Sci. Eng. (CSE)*, Bucharest, Romania, Oct. 2018, pp. 194–200, doi: 10.1109/CSE.2018.00034.

[30] Z. Fan and R. Liu, "Investigation of machine learning based network traffic classification," in *Proc. Int. Symp. Wireless Commun. Syst. (ISWCS)*, Bologna, Italy, Aug. 2017, pp. 1–6, doi: 10.1109/ISWCS.2017.8108090.

[31] Y. Chen, J. Pei, and D. Li, "DETPro: A high-efficiency and low-latency system against DDoS attacks in SDN based on decision tree," in *Proc. ICC-IEEE Int. Conf. Commun. (ICC)*, Shanghai, China, May 2019, pp. 1–6, doi: 10.1109/ICC.2019.8761580.

[32] C.-C. Liu, Y. Chang, C.-W. Tseng, Y.-T. Yang, M.-S. Lai, and L.-D. Chou, "SVM-based classification mechanism and its application in SDN networks," in *Proc. 10th Int. Conf. Commun. Softw. Netw. (ICCSN)*, Chengdu, China, Jul. 2018, pp. 45–49, doi: 10.1109/ICCSN.2018.8488219.

[33] M. Balta and I. Özçelik, "A 3-stage fuzzy-decision tree model for traffic signal optimization in urban city via a SDN based VANET architecture," *Future Gener. Comput. Syst.*, vol. 104, pp. 142–158, Mar. 2020.

[34] R. Santos, D. Souza, W. Santo, A. Ribeiro, and E. Moreno, "Machine learning algorithms to detect DDoS attacks in SDN," *Concurrency Comput., Pract. Exper.*, vol. 32, no. 16, pp. 1–14, Jun. 2019.

[35] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proc. 6th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, Boston, MA, USA, 2000, pp. 71–80, doi: 10.1145/347090.347107.

[36] W. Hoeffding, "Probability inequalities for sums of bounded random variables," in *The Collected Works of Wassily Hoeffding*. New York, NY, USA, Springer, 1994, pp. 409–426, doi: 10.1007/978-1-4612-0865-5_26.

[37] M. H. H. Khairi, S. H. S. Ariffin, N. M. A. Latiff, and K. M. Yusof, "Generation and collection of data for normal and conflicting flows in software defined network flow table," *Indonesian J. Electr. Eng. Comput. Sci.*, vol. 22, no. 1, p. 307 314, Apr. 2021.

[38] V. Danciu and C. N. Tran, "Side-effects causing hidden conflicts in software-defined networks," *Social Netw. Comput. Sci.*, vol. 1, no. 5, pp. 1–16, Aug. 2020.

[39] S. Pisharody, J. Natarajan, A. Chowdhary, A. Alshalan, and D. Huang, "Brew: A security policy analysis framework for distributed SDN-based cloud environments," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 6, pp. 1011–1025, Nov. 2019, doi: 10.1109/TDSC.2017.2726066.

[40] H. Hu, W. Han, G.-J. Ahn, and Z. Zhao, "FLOWGUARD: Building robust firewalls for software-defined networks," in *Proc. 3rd Workshop Hot Topics Softw. Defined Netw.*, Chicago, IL, USA, Aug. 2014, pp. 97–102.

MUTAZ HAMED HUSSIEN KHAIRI received the B.S. degree in computer engineering from Future University, in 2002, and the M.S. degree in electrical engineering from Linkoping University, in 2007. He is currently pursuing the Ph.D. degree with Universiti Technologi Malaysia (UTM). From 2002 to 2007, he worked as a Lecturer with the Faculty of Engineering, Future University, and the Director of Information Technology Department. He is currently a Researcher and an ICT Specialist with 16 years of wide range of research and ICT experience. His main research interests include software define networks (SDNs), machine learning, telecommunication networks, and antenna design and implementation.

SHARIFAH HAFIZAH SYED ARIFFIN (Senior Member, IEEE) received the B.Eng. degree (Hons.) from London, in 1997, the M.E.E. degree from Universiti Teknologi Malaysia, in 2001, and the Ph.D. degree from Queen Mary, University of London, London, in 2006. She is currently an Associate Professor with the Faculty of Electrical Engineering, Universiti Teknologi Malaysia. She had published 116 articles, 17 copyrights, one integrated circuit, and one trademark. Her current research interests include the Internet of Things, ubiquitous computing and smart devices, wireless sensor networks, ipv6, handoff management, networks, and mobile computing systems.

NURUL MU'AZZAH ABDUL LATIFF (Senior Member, IEEE) received the B.Eng. degree in electrical-telecommunications from Universiti Teknologi Malaysia, in 2002, the M.Sc. degree in communications and signal processing from Newcastle University, U.K., in 2003, and the Ph.D. degree in wireless telecommunication engineering, in 2008. Since 2002, she has been with the School of Electrical Engineering, Universiti Teknologi Malaysia, Johor Bahru, Malaysia, where she is currently an Associate Professor. She is also a Chartered Engineer with the Institution of Engineering and Technology (IET). Her research interests include wireless networks, ranging from theory, to design and implementation. In particular, she has great interest in wireless sensor networks, mobile *ad hoc* networks, cognitive radio networks, the Internet of Things, and optimization of wireless network based on artificial intelligence and machine learning algorithm for healthcare applications. She is an Active Volunteer with the IEEE Communication/Vehicular Technology Society, Malaysia.

KAMALUDIN MOHAMAD YUSOF received the B.Eng. degree in electrical-electronics engineering and the M.Eng. degree in electrical engineering from Universiti Teknologi Malaysia, and the Ph.D. degree from University of Essex, U.K. He is currently a Senior Lecturer with the Division of Communication Engineering, School of Electrical Engineering, Faculty of Engineering, Universiti Teknologi Malaysia. His current research interests include the Internet of Things, big data, and software-defined networks. He is also a member of the Communication Network System (CNetS).

MOHAMED KHALAFALLA HASSAN received the B.Sc. degree in computer engineering from Future University Sudan, in 2004, and the M.Sc. degree in communication network engineering from Universiti Putra Malaysia (UPM), in 2009. He is currently pursuing the Ph.D. degree in communication engineering with Universiti Technology Malaysia (UTM). He is currently a Researcher and an ICT Specialist with 16 years of wide range of research and ICT experience. He has 15 article published in international peer reviewed conferences and journals. His main research interests include forwards scattering radar, machine learning, nfv, vsdn, and resources management in communication networks.

FAHAD TAHA AL-DHIEF (Graduate Student Member, IEEE) was born in Amarah, Iraq. He received the B.S. degree in software engineering from Imam Ja'afar Al-Sadiq University, Iraq, in 2013, and the M.S. degree in computer science from Universiti Kebangsaan Malaysia, Malaysia, in 2016. He is currently pursuing the Ph.D. degree with the School of Electrical Engineering, Universiti Teknologi Malaysia, Malaysia. His research interests include sensor networks, routing protocols, mobile *ad hoc* networks, social networks, the Internet of Things, machine learning, artificial neural networks, deep learning, and location-based service. He is a member of the IEEE Communications Society.

MOSAB HAMDAN (Graduate Student Member, IEEE) received the B.Sc. degree in computer and electronic system engineering from the University of Science and Technology (UST), Sudan, in 2010, the M.Sc. degree in computer architecture and networking from the University of Khartoum (UofK), Sudan, in 2014, and the Ph.D. degree in electrical engineering (computer networking) from the Faculty of Engineering, School of Electrical Engineering, Universiti Teknologi Malaysia (UTM), Malaysia, in 2021. From 2010 to 2015, he was a Teaching Assistant and a Lecturer with the Department of Computer and Electronics System Engineering, Faculty of Engineering, University of Science and Technology (UST). He is currently a Researcher with the Universiti Teknologi Malaysia under the postdoctoral fellowship scheme. His current research interests include software-defined networking (SDN), load balancing, network traffic classification, the Internet of Things (IoT), cloud computing, network security, and future networks.

SULEMAN KHAN received the Ph.D. degree (Hons.) in computer science and information technology from Universiti Malaya, Malaysia, in 2017. He was a Faculty Member with the School of Information Technology, Monash University, Malaysia, from June 2017 to March 2019. He is currently a Faculty Member with the Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne, U.K. He has published more than 60 high-impact research articles in reputed international journals and conferences. His research interests include network forensics, software-defined networks, the Internet of Things, cloud computing, and vehicular communications.

MUZAFFAR HAMZAH received the B.Sc. and M.Sc. degrees in computer science from the University of Malaya, and the Ph.D. degree in information technology from the University of South Australia. He is currently the Deputy Dean (graduate and internationalization) and a Senior Lecturer with the Faculty of Computing and Informatics, Universiti Malaysia Sabah, Malaysia. He received a Postdoctoral Research in geovisualization with Nottingham University. He actively involved in research about HCI and information visualization. His research interests include information theory, visual analytics, and formal methods. He has also led several research grants at national level as a principal investigator. He received the Best Paper Award from IBIMA.