A REVIEW OF FLOW CONFLICTS AND SOLUTIONS IN SOFTWARE DEFINED NETWORKS (SDN)

MUTAZ HAMED HUSSIEN KHAIRI[1,2*], SHARIFAH HAFIZAH SYED ARIFFIN[1], NURUL MUAAZAH. ABDUL LATIFF[1],
KAMALUDIN MOHAMED YUSUF[1] AND MOHAMED KHALAFALLA HASSAN[1,2]

[1]School of Electrical Engineering, Faculty of Engineering,
University Technology Malaysia, Johor, Malaysia
[2]Faculty of Engineering, Future University, Khartoum, Sudan

[*]Corresponding author: taza1040@gmail.com

(Received: 17th September 2020; Accepted: 16th February 2021; Published on-line: 4th July 2021)

ABSTRACT: Software Defined Networks (SDN) are a modern networking technology introduced to simplify network management via the separation of the data and control planes. Characteristically, flow entries are propagated between the control plane layer and application or data plane layers respectively while following flow table instructions through an OpenFlow protocol. More often than not, conflicts in flows occur as a result of traffic load and priority of instructions in the data plane. Several research works have been conducted on flow conflicts in SDN to reduce their adverse effect. Solutions to flow conflict in SDN have three main limitations. First, the OpenFlow table may still cause a defect in the security module according to the priority and action matching in the OpenFlow of the control plane. Second, flow conflict detection requires more time due to flow tracking and incremental update, whereas in such a case, delay affects the efficiency of SDN. Besides, the SDN algorithm and mechanism have substantially high memory requirement for instruction and proper functioning. Third, most of the available algorithms and detection methods used to avoid flow conflicts have not fully covered the security model policy. This study reviews these limitations and suggest solutions for future research directions.

ABSTRAK: Rangkaian Perisian Tertentu (SDN) adalah teknologi rangkaian moden yang diperkenalkan bagi memudahkan pengurusan rangkaian melalui pecahan data dan kawalan permukaan. Seperti biasa, aliran kemasukan disebar luas antara lapisan permukaan kawalan dan aplikasi atau lapisan permukaan data masing-masing, sambil mengikuti arahan meja melebar melalui protokol aliran terbuka. Kebiasaannya konflik dalam aliran berlaku disebabkan oleh beban trafik dan keutamaan arahan pada permukaan data. Beberapa kajian dibuat terhadap konflik aliran SDN bagi mengurangkan kesan konflik. Solusi konflik aliran dalam SDN mempunyai tiga kekurangan besar. Pertama, jadual Aliran Terbuka mungkin masih menyebabkan kekurangan dalam modul keselamatan berdasarkan keutamaan dan tindakan persamaan dalam aliran terbuka permukaan kawalan. Kedua, pengesanan aliran konflik memerlukan lebih masa bagi pengesanan aliran dan peningkatan kemaskini, kerana setiap penangguhan memberi kesan terhadap kecekapan SDN. Selain itu, algoritma SDN dan mekanisme memerlukan memori yang agak besar bagi memproses arahan dan berfungsi dengan baik. Ketiga, kebanyakan algoritma dan kaedah pengesanan yang digunakan bagi mengelak konflik pengaliran tidak sepenuhnya dilindungi polisi model keselamatan. Oleh itu, kajian ini meneliti kekurangan dan memberi cadangan penambahbaikan bagi arah tuju kajian masa depan yang terbuka.

KEYWORDS: *software defined networking; open flow table; flow conflicts*

1. INTRODUCTION

Software Defined Network (SDN) is a modern network architecture that has been proposed to mitigate the shortcomings related to traffic management, security, virtualization and complexity of traditional networks [1]. The main characteristics of SDN include: 1) separation and abstraction of the control and data plane; 2) logical centralization of network intelligence that enables global view of the entire network and swift response changing needs; 3) ability to develop different applications from the underlying network infrastructure; 4) programmable data plane for networking automation and flexibility; 5) much higher innovation speed to accelerate business innovation and possibility of reprograming of the network by the IT network operators in real time to meet specific business needs. Moreover, through the virtualization process, the network infrastructure can be extracted from individual network services [2].

OpenFlow is a SDN protocol designed to facilitate server communication with network switches; particularly regarding sending and receiving packets. As a flow-based network virtualization architecture, it allows multiple logical networks to share the same physical infrastructure. While the network virtualization layer allows for a set of controllers to manage multiple switches per slice, controllers are responsible for installing flow entries in the assigned domain of switches. In such a design, one physical switch could belong to multiple virtual networks which may be controlled by one or more set of controllers. This design setup leads to flow conflicts [3]. In today's Internet, network traffic is routed based on the destination address prefixes. While this approach allows an efficient implementation of shortest-path (and more complex) routing protocols, it does not provide fine-grained control over network traffic. However, many proposals for future internet architectures require that the network data plane implement routing and forwarding at the level of individual connections or their aggregate e.g., for network virtualization, or for network services [4,5].

2. PROBLEM BACKGROUND AND OBJECTIVES OF THE RESEARCH

Conflict in flow entries manifest in different ways in SDN such as where a physical switch belongs to multiple virtual networks that are often controlled by one or a set of controllers; thus, resulting in the occurrence of flow conflict. Flow conflicts may also occur when vague packets match flow entries. Based on the priority of the flow entry, flow entry conflicts in an open flow table can cause defects in the security module that runs on top of the SDN. An approach used, is to match flow entry packets based on the priority of flow entries. However, because of the similarities between different flow entries, a single packet can be matched to several flow entries, thereby resulting in flow entry conflict [1].

While representative algorithms and detection methods have been proposed to mitigate flow entry conflicts [1-3], they have been shown to have high memory requirements and require a significant amount of time to apply their instructions in the flow table. Specifically, the time taken to add or update flows in case of conflict in the flow entry is high; thus, affecting the performance of the SDN network. In addition, most of these past studies used old versions of flow table in their simulations and experiments.

The present study was initiated,

i. To study and analyze flow entry conflicts in OpenFlow table.

ii. To review the existing detection methods used for flow conflict resolution in OpenFlow table.

3. SDN ARCHITECTURAL COMPONENTS

The three core components of the SDN architecture are: the SDN controller, also known as control layer; the data plane layer, which is made up of a collection of network devices like routers, switches and OpenFlow switches that act as middle boxes for network communication; and the application layer, where the execution of all network applications take place. Figure 1 shows the SDN architecture. It can be clearly observed that it is characterized by the separation of the control and infrastructure layers. This is advantageous in that the application layer can be developed and modified by application developers by using a northbound interface that provides advanced policy applications and services as well as programmable Application Programming Interfaces (API) for this purpose. Moreover, the southbound interface also provides the OpenFlow protocol, which is a standard API [4,5].

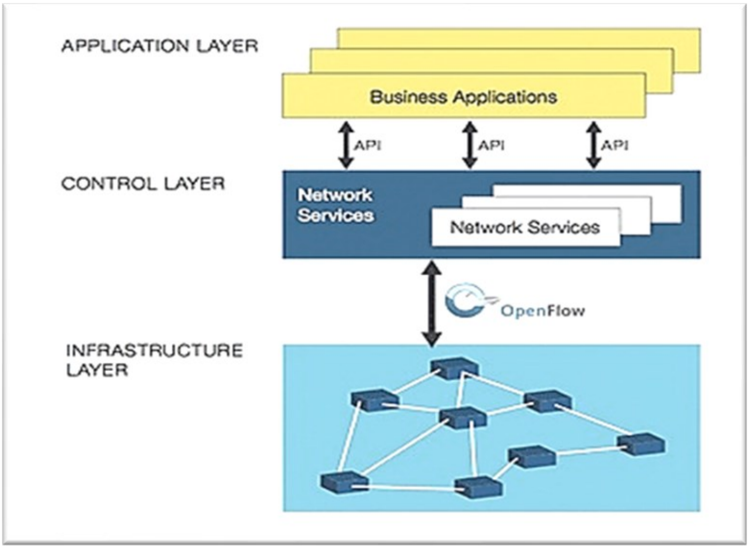

Fig. 1: Software Defined Networking framework [6].

4. OPENFLOW (OF) IN SDN

OpenFlow was one of the first software defined networking (SDN) standards. It was originally a communication and connection protocol in SDN environments that allowed the controller to directly interact with the forwarding plane of network devices for better adaptation to changing business requirements [7].

4.1 OpenFlow Switch and Protocol

OpenFlow was driven by the characteristic separation of the control and forwarding planes in SDN. The OpenFlow switch architecture is illustrated in Fig. 2. The OpenFlow switch, which also contains a number of flow tables, is connected to the controller via an OpenFlow channel. Likewise, OpenFlow switches also have an abstraction layer through which they communicate with the controller via the OpenFlow protocol. The flow table checks which packet belongs to which flow in order for the packets to be processed and delivered [1,4].

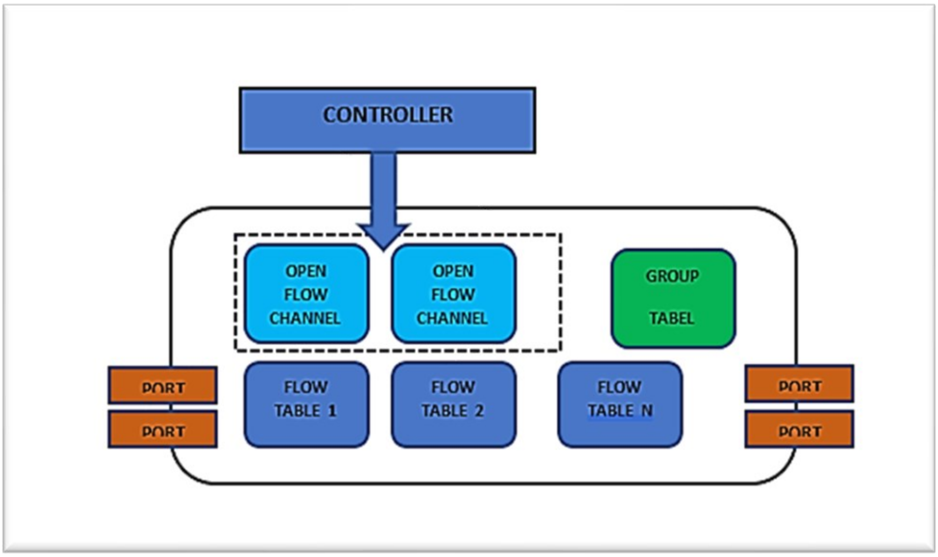

Fig. 2: OpenFlow switch [5].

- Group table: for packet lookup and forwarding.
- OpenFlow channel: responsible for connection between controller and switch.
- Flow table: flow entry – match field – counter and set instruction.

4.2 OpenFlow Ports

OpenFlow ports perform the function of transferring packets between open flow switches and other devices in the network such as routers or hosts. All OpenFlow switches in the SDN network are connected through OpenFlow ports. Some network interfaces may also be disabled in the OpenFlow switch [8].

4.3 OpenFlow Table and Entry

The OpenFlow switch is comprised of several flow tables, each of which has flow entries. The processing pipeline of OpenFlow stipulates the interaction between packets and flow tables. Depending on the application and network structure, an OpenFlow switch may have one or more flow tables [8].

4.4 Flow Table

Flow tables consist of several flow entries as shown in Table 1.

Table 1: Flow entries and their definitions [8].

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie
Used to match between packets	For matching priority of flow entries	To update the proper packets	To adjust the action set of a flow entry	Maximum time or inactive time before switch terminates a flow entry	For reporting a modified or deleted flow entry to controller

5. EXISITING SOLUTIONS TO FLOW CONFLICTS IN SDN

This section discusses the research that has been conducted on detecting flow conflicts in SDN. The studies are explained based on their method, controller type, switch, conflict type, topology, and target application respectively, as in Table 2.

Table 2: List of Existing Solution on Flows Conflict in SDN

Method or Techniques	Reference Research	Controller Model	Switch Model	Conflict type	Network topology	Target application
Mechanism or Algorithm	Lo et al. [1]	-	-	Flow entries	Single topology	Network optimization performance
	Hu et al. [3]	Floodlight	OpenFlow Switch	Flow rule	Tree topology	Security
	Fang and Lu [7]	-	-	Flow rule	Single topology	Network efficiency
	Pisharody et al. [9]	Open Daylight	OpenFlow switch	Flow rule	Tree topology	Security
	Lu et al. [10]	NOX	OpenFlow Switch	Policy rule	Tree topology	Security
	Wang and Youn [11]	Floodlight	Open vSwitch	Flow entries	Single topology	Network optimization performance
	Cui et al. [12]	Floodlight	OpenFlow Switch	Flow rule	Tree topology	Network efficiency
	Hao et al. [13]	NOX	OpenFlow switch	Flow rule	Tree topology	Network optimization performance
	Halder et al. [14]	OF Controller	OpenFlow switch	Flow rule	Tree topology	Security
	Batista et al. [15]	OF Controller	OpenFlow switch	Flow entries	Single topology	Security
	Hong and Wey [16]	Floodlight	Open vSwitch	Flow rule	Tree topology	Network optimization performance
Analytical method	Metter et al. [2]	-	-	Flow rule	Single topology	Network optimization performance
	Lin et al. [17]	Ryu	OpenFlow switch	Flow entries	Tree topology	Network optimization performance
	Tran and Danciu [18]	-	-	Flow rule	Tree topology	Network optimization performance
	Yoshioka et al. [19]	-	-	Flow rule	Single topology	Link utilization

5.1 Purpose of Flow Conflict Detection Techniques

Flow entry conflict detection methods in SDN applications have been used for different purposes such as security, load balance, and firewall amongst others, as shown in Table 2. A number of detection algorithms have attempted to optimize the performance of the controller by checking and arranging its workload. In [16], the controller was used to control elephant and safety-critical flows using interactive routing. In the study, switch layers were designed to move micro flows through multipath wildcard routing without soliciting or using the controller plane. A firewall policy, named Flow Guard, that is meant to secure the policy contravention in flow table of the OpenFlow switch with controller was proposed in [3]. It tracks flow or path spaces in the network to detect firewall policy contravention within network updates. In a separate work in [20], a method was proposed to check and detect contradiction between the rules installed in a firewall with flow table instructions. The rules applied in the OpenFlow design is to increase the safety and security capabilities for SDN.

5.2 Method and Algorithm Used for Flow Conflict Detection

One of the famous approaches used in the detection of conflicts in flow entries within SDN is the Reduced Bit Vector algorithm. This algorithm uses the concept of bit vectors alongside one of the classification methods to classify flow entries in two main groups. In the first group, flows have the same prefix length while the second group contains a decrease in the redundant bits of vectors [1]. One of the solutions introduced and built for optimizing and securing flow entry in SDN research is a security module that minimizes the amount of controller workload by separating the responsibilities of each controller [16]. A simple analytical model is formulated to enable network performance optimization relative to the signaling rate and table occupancy [2]. The research used a flow guard method for building a firewall to conduct checks and detect violation of firewall policy along the network flow path when updating the status of the network [3]. A routing method is used to ensure optimal link utilization as well as in the minimization of flow entry size. To ensure optimal link utilization, the proposed method assigns the same paths to flows that can be grouped together [19]. A security application, named FRESCO, has been proposed and implemented to address security challenges. The FRESCO framework consists of an application layer and a security enforcement kernel [21] Brew. The proposed policy is used to analyze the controller settings to detect conflicts and create solution modules which avoids conflict in flow rules within a distribution-based SDN system [9]. Source routing has also been used in flow route control. In [22], a Source Routing Based Link Protection Method whose link has paths to dedicated backup was used to direct packets to their backup paths by updating the header of the source router in case of link failure. Performance evaluation of this method has been shown to significantly update the source routing header.

5.3 Experimental Environment Used in Flow Conflict Detection

The experimental environment used for flow entry conflict detection in SDN varies according to the method or algorithm. This section details the experimental environments used in flow conflict detection studies in SDN. [16] developed a simple network with four switches (grid network) connected to one host. Using a floodlight controller with OpenFlow switch version 1.3, all tests and observations were simulated and analyzed with Mininet software. Many studies in SDN use MATLAB software with Mininet and virtual machine software as simulation tools for analyzing and optimizing the traffic and flow entry in SDN. Figure 3 shows an example of a similar research, where the design structure depends on how the OpenFlow switch deals with controller under FRESCO instruction [2].

Additionally, Fig. 4 shows the Brew system model. It uses two virtual switches connected to a controller where all the conflicting flow entries are checked and analyzed in line with the controller plane [9].

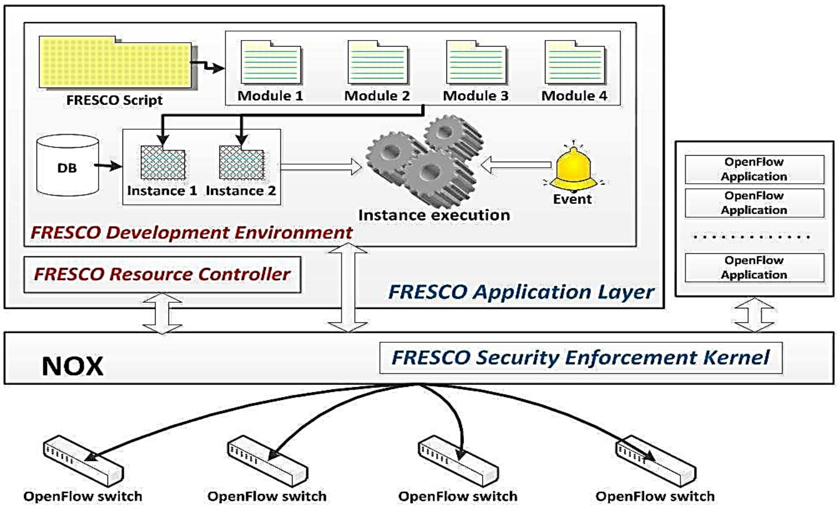

Fig. 3: High level overview of the FRESCO architecture [2].

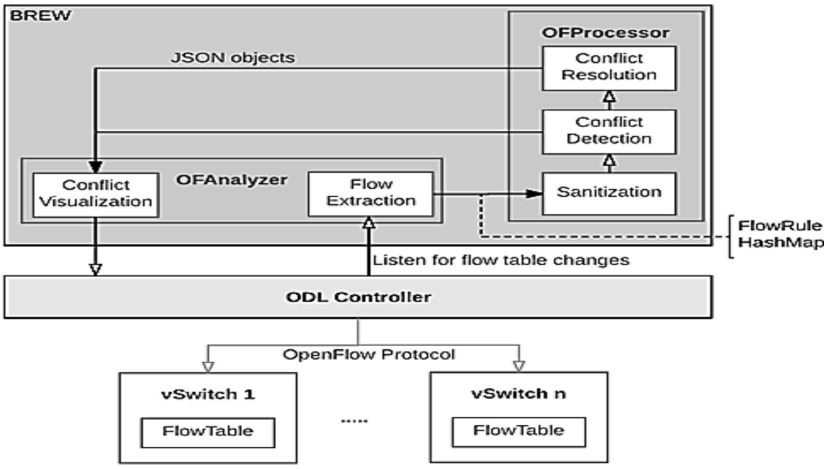

Fig. 4: System architecture showing Brew solution models [9].

5.4 SDN Controller Used in Flows Conflict Detection Techniques

The SDN controller is called the “network brain” [23]. It ensures that different network tasks are performed through effective management of the application modules of the network. The applications typically leverage an API, as in [24], to enable advanced network functionalities while facilitating communication with core controller modules. We discuss both commercial and open-source controllers that are widely used in load balancing modules of the server. While the software load balancing module in the commercial controllers are standalone systems connected to the controller for API utilization purposes, they function as part of the controller system within the application layer in the open-source version. A popular open-source controller is the high-performance java-based OpenFlow controller called Flood Light Controller, which is based on Beacon controller, an experimental OpenFlow controller from Stanford University that now has a large community of developers that supports it. The latest version of Floodlight has support for OpenFlow version 1.4 [25]. Open Daylight (ODL) SDN controller is another type of controller used to implement Brew [9].

5.5 Parameter Measure in Flow Conflict Detection

Flow entry in SDN is generally measured by different types of parameters, we focus only on parameters such as time of search, update time, elephant flows, and Link Bandwidth as measured parameters in the flow entry conflict detection.

6. ANALYSIS AND DISCUSSION

We aim, by this research, to define the types of flow rule conflict that occur in SDN. A number of researchers have used a Static Programming language module by leveraging the OF Analyzer module named Brew; a security policy analysis framework that is developed on an Open Daylight (ODL) based SDN controller [9]. In [10], a method was proposed for accurate detection of universal conflicts. In the proposed mechanism, checks are carried out for policy conflict with machine learning for 500 policies applied in a simulated network. The study also introduced a novel flow management project for MFTs for Open vSwitch in SDN. The memory requirement of the research shows that the Ex table requires a large memory space [11].

The authors of [7] used the MTBDD algorithm to conduct and clarify switch conflict rules in SDN. The test of algorithm offers good functionality on incremental updates of two different rule groups from Stanford's Internet network. In a separate research, TCDR was used to reject illegal flow rules, thus, avoiding substantial conflicts in flow rules while incurring minimal overhead [12].

The detection mechanism for conflicting flows in the rules of a flow table as presented in [13] takes 1ms of time. However, it should be noted that the effect of the firewall rule is not considered for table update. In [14], a graph based technique is applied to produce a guide graph from obtainable flow rules built-up by several controller applications within SDN to determine flow transgression. A classification method is proposed to classify the flow entries in two main groups; with the first group of the flow having same prefix length, while the second group contains a decrease in the redundant bits of vectors . In using RVB, there is still notable delay in time of search and incremental update (over 2 ms) [1]. In [15], the author used an intelligent technique for conflict in flow entries within SDN. While the proposed method effectively detected most of the conflicts that appeared in the flow table, its high memory requirement remained a major shortcoming.

Lin et al. [17] used most of the parameters configured in OpenFlow tables to conduct cross-layer testing between two different flow entries. The priority of flow of the entire inflow table was experimentally observed. The proposed method showed impressive accuracy. The research analytically showed that the conflict of applications in SDN can be detected and analyzed without stopping the application [18].

An analytical model has been proposed for SDN Signaling Traffic and Flow Table Occupancy in [2]. The model was designed to allow network performance optimization relative to signaling rate and table occupancy. The importance of this model could decrease in the near future when switches are upgraded or the size of the flow table is increased. A method of optimizing controller workload with a view to minimizing the amount of controller workload by separating the responsibility of each controller has been demonstrated in [16]. The potential services considered and checked in this research do not include File Transfer Protocol (FTP).

A method, called flow entry aggregation, capable of reducing the maximum number of flow entries while suppressing the maximum link utilization, was proposed in [20].

However, a key drawback of this method is that it cannot by itself prevent any loops from occurring in the network; thus, it is susceptible to flow entry conflicts [19].

7. CONCLUSION

What has been presented in this research is mainly concerned with flow conflict issues in SDN and how they have been mitigated in the flow table. According to the reviewed and discussed articles, flow conflicts can be classified into two major types, flow rules and flow entries. Summarily, researchers have used only two methods to detect flow conflict in SDN; majority of them using an algorithmic approach while a few used analytical methods. Floodlight, Open Daylight, POX and Ryu controllers were the most commonly used controllers in experimental and test studies while only two types of network topologies, tree and single topology, were used in the pertinent literatures. The target application in most studies is network optimization and performance while only a few studies targeted security issues.

From all the reviewed studies, a research involving the use of Artificial Intelligence such as the machine learning method to detect flow conflicts in the OpenFlow table for network security related issues still remain lacking. Thus, this study proposes an anomaly detection algorithm using machine learning to detect and classify flow conflicts in the OpenFlow table as an interesting future research direction. The goals of such an algorithm will be to classify flow rules while concurrently reducing the delay in the update flow table of the controller.

REFERENCES

[1] Lo C, Wu P, Kuo Y. (2015) Flow entry conflict detection scheme for software-defined network. International Telecommunication Networks and Applications Conference (ITNAC), Sydney, Australia, pp. 220-225.

[2] Metter C, Seufert M, Wamser F, Zinner T, Tran-Gia P. (2017) Analytical model for SDN signaling traffic and flow table occupancy and its application for various types of traffic. IEEE Transactions on Network and Service Management, 14(3): 603-615.

[3] Hu H, Han W, Ahn GJ. Zhao Z. (2014) FLOWGUARD: Building robust firewalls for software-defined networks. in Proceedings of the third workshop on Hot topics in software defined networking August 2014. pp 97-102.

[4] Akyildiz, IF, Lee A., Wang P, Luo M, Chou W. (2014) A roadmap for traffic engineering in SDN-OpenFlow networks. Computer Networks, 71: 1-30.

[5] Bozakov Z, Sander V. (2013) OpenFlow: A perspective for building versatile networks. in: Network-Embedded Management and Applications, Clemm A, Wolter R. (eds). Springer, New York, NY. https://doi.org/10.1007/978-1-4419-6769-5_11

[6] sdx central [https://www.sdxcentral.com/networking/sdn/definitions/what-the-definition-of-software-defined-networking-sdn]

[7] Fang Y, Lu Y. (2019) Checking intra-switch conflicts of rules during preprocessing of network verification in SDN. IEEE Communications Letters, 23(9): 1547-1550.

[8] OpenFlow Specification [http://networkstatic.net/wp-content/uploads/2013/02/openflow-spec-v1.3.0.pdf]

[9] Pisharody S, Natarajan J, Chowdhary A, Alshalan A, Huang D. (2019) Brew: A security policy analysis framework for distributed SDN-based cloud environments. IEEE Transactions on Dependable and Secure Computing, 16(6): 1011-1025.

[10] Lu Y, Fu Q, Xi X, Chen Z, Zou E, Fu B. (2019) A policy conflict detection mechanism for multi-controller software-defined networks. International Journal of Distributed Sensor Networks 15 No (5),http://doi.org/10.1177/1550147719844710.

[11] Wang C, Youn HY. (2019) Entry aggregation and early match using hidden Markov model of flow table in SDN. Sensors, 19(10), 2341; https://doi.org/10.3390/s19102341.

[12] Cui J, Zhou S, Zhong H, Xu Y, Sha K. (2018) Transaction-based flow rule conflict detection and resolution in SDN. 27th International Conference on Computer Communication and Networks (ICCCN), Hangzhou, China, pp. 1-9.

[13] Hao W, Jiang Y, J. Gao J. (2017) Detection mechanisms of rule conflicts in SDN based on a path-tree model. 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, pp. 336-339.

[14] Halder B, Barik MS, Mazumdar C. (2017) A graph based formalism for detecting flow conflicts in software defined network. 2017 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), Bhubaneswar, India, pp. 1-6.

[15] Lopes Alcantara Batista B, Lima de Campos GM, Fernandez MP. (2014) Flow-based conflict detection in OpenFlow networks using first-order logic. 2014 IEEE Symposium on Computers and Communications (ISCC), Funchal, Portugal, pp. 1-6.

[16] Hong ETB, Wey CY. (2017) An optimized flow management mechanism in OpenFlow network. 2017 International Conference on Information Networking (ICOIN), Da Nang, Vietnam, pp. 143-147.

[17] Lin YD, Lai YK, Tsou YL, Lai YC, Liou EC, Chiang Y. (2019) Generic validation criteria and methodologies for SDN applications. IEEE Systems Journal, 13(4): 3909-3920.

[18] Tran CN, Danciu, V. (2020) A general approach to conflict detection in software-defined networks. SN Comput. Sci, 1(1), 9, https://doi.org/10.1007/s42979-019-0009-9

[19] Yoshioka K, Hirata K, Yamamoto M. (2017) Routing method with flow entry aggregation for software-defined networking. 2017 International Conference on Information Networking (ICOIN), Da Nang, Vietnam, pp. 157-162,

[20] Pallavi N., Anisha A.S., Leena V. (2017) Detection of Incongruent Firewall Rules and Flow Rules in SDN. In: Dash S., Vijayakumar K., Panigrahi B., Das S. (eds) Artificial Intelligence and Evolutionary Computations in Engineering Systems. Advances in Intelligent Systems and Computing, vol 517. Springer, Singapore. https://doi.org/10.1007/978-981-10-3174-8_2.

[21] Shin SW, Porras P, Yegneswara V, Fong M, Gu G, Tyson M. (2013) Fresco: Modular composable security services for software-defined networks. in February 2013- 20th Annual Network & Distributed System Security Symposium. http://hdl.handle.net/10203/205914

[22] Huang L, Shen Q, Wenjuan S. (2016) A source routing based link protection method for link failure in SDN. 2nd IEEE International Conference on Computer and Communications (ICCC), Chengdu, China, pp. 2588-2594.

[23] Jarschel M, Zinner T, Hoßfeld T, Tran-Gia P, Kellerer W. (2014) Interfaces, attributes and use cases: A compass for SDN. IEEE Communications Magazine, 52(6): 210-217.

[24] Zhou, J. (2014) Multicatalyst system in asymmetric catalysis. John Wiley & Sons.

[25] OpenFlow Switch Specification OpenFlow Switch Specification, O.S., Version 1.4. 0, October 14, 2013. [https://opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.4.0.pdf]