Egyptian Informatics Journal 23 (2022) 149–157

Contents lists available at ScienceDirect

Egyptian Informatics Journal

journal homepage: www.sciencedirect.com

Automised flow rule formation by using machine learning in software defined networks based edge computing

Saleem Iqbal [a], Hira Maryam [b], Kashif Naseer Qureshi [a], Ibrahim Tariq Javed [c,*], Noel Crespi [d,e]

[a] University Institute of Information Technology, Arid University Rawalpindi, Pakistan
[b] Department of Software Engineering, Bahria University, Islamabad, Pakistan
[c] Department of Computer Science, Bahria University, Islamabad 44000, Pakistan
[d] Lero-Science Foundation Ireland Research Centre for Software, University of Limerick, V94 T9PX Limerick, Ireland
[e] Institut Polytechnique de Paris Telecom SudParis Evry, Courcouronnes FR, 9 Rue Charles Fourier, 91000, France

ARTICLE INFO

Article history:
Received 3 August 2021
Revised 3 October 2021
Accepted 16 October 2021
Available online 5 November 2021

Keywords:
Software-Defined-Network Controller
Machine Learning
Flow Rule
OpenFlow
Auto Rule Formation

ABSTRACT

The availability of Software Defined Network's (SDNs) flow rule entry in the flow table is considered a key factor in the timely delivery of a certain flow. The controller is approached for instructions on how to deal with the flow when the rule for such flow is missing. The controller then updates the flow table accordingly at the switch so that flow could be dealt with locally. It becomes problematic when no rule is defined yet at the controller by the application plane. In most of these cases, such a situation is handled by programming the controller with wildcard rules. However, handling many flows at once under wildcard rules severely hampers the network performance. Flow rules formation by the application plane is sometimes critical and time-consuming which increases the latency ratio by creating a bottleneck at the switch level. To avoid the bottlenecks due to rule absence, in this paper, rather than waiting for the application plane's response and putting the pending traffic flows in the buffer, which may be dropped, the controller is programmed in a way that has the built-in mechanism of self-flow rule formation. This atomized mechanism is based on the previously available traces of the same flows when they were forwarded on the network using the wildcard rules. To assess the performance of the proposed work, it is emulated and benchmarked with the latest research. The results show considerable performance achievement.

1. Introduction

In the past few years, Software Defined Networking (SDN) has gained popularity in large organizations that are shifting from traditional systems to SDN based organized and controlled systems. In SDN, the control plane is separated from the data plane and divided the complete paradigm into two planes which do not only provide efficiency but also provide reliability. The SDN networks are based on three layers including the application layer, control layer, and data layer. These three layers are connected through the Application Programming Interface (API). The application layer is connected to the control layer through the northbound API and on the other side controller is connected to the data layer through the southbound API [1–3]. SDN is more reliable, efficient, and more secure than the traditional networks because of its separation strategy of control plane and data plane. In SDN, OpenFlow protocol is used at both ends of the controller for other layers' interaction and processes. Whenever a switch sends a request for the connection it uses a request to runs the Transmission Control Protocol (TCP) [4–6]. There is a flow table within a switch that defines what actions should be applied to the packets in SDN by specifying the match criteria and actions. Match criteria include IP source/destination and protocol or maybe other headers as well. Actions are related to dropping a packet or sending a packet to the controller. For example, the source TCP is 10.0.0.100, and the destination IP is 10.0.0.2, when the packet arrives on the system it

* Corresponding author.
E-mail addresses: saleem@uaar.edu.pk (S. Iqbal), knaseer.buic@bahria.edu.pk (K.N. Qureshi), noel.crespi@mines-telecom.fr (N. Crespi).

Peer review under responsibility of Faculty of Computers and Information, Cairo University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.eij.2021.10.001
1110-8665/

matches against the entries in the table. This packet matches to destinations when the protocol is TCP, but the source and destination of the packet are not specified. The switch determines the rule to given priority, so the priority field is also the part of the flow table in case if any packet matches with two destinations at the same time [7]. A larger number refers to the larger priority in the table. The hard timeout field is also part of the flow table. Hard time outfield works after 20 s and the rule is removed by the switch automatically if hard time out is 20 s. If a hard time is set on zero, the rule remains there and is not removed automatically. Idle time out is also an entity of the rule table if no packet matches with any entry in the table for the given amount of time and removed. In case a packet arrives and it does not match with any of the entry for example src: 10.0.0.100 DST: 10.0.0.3 switch will send this particular packet to the controller, the controller has a control channel that has a TCP connection with each switch in the data plane the core matches each TCP connection and creates a switch added event [8–10]. App sends a flow mod to install a switch and when a packet does not match with any entry in the table then a packet sent to the controller and the application installs the rules in the flow mod and removes rules. It also includes topology discovery and special control packets. When a switch receives one of these packets, it notifies the topology discovery and creates an event, and sends this event to the application. Whenever a packet arrives packet entries are tested with the entries of the corresponding rule table to check which rule goes for which packet, when all the entries are matched then the rule is applied to that packet. If the matching rule is not to find, then the controller sends a control message to the application layer to get the rules for the specific packet. The same process is repeated if any new entry is found and can also be added to the flow table [11,12]. The structure of the flow table is shown in Fig. 1.

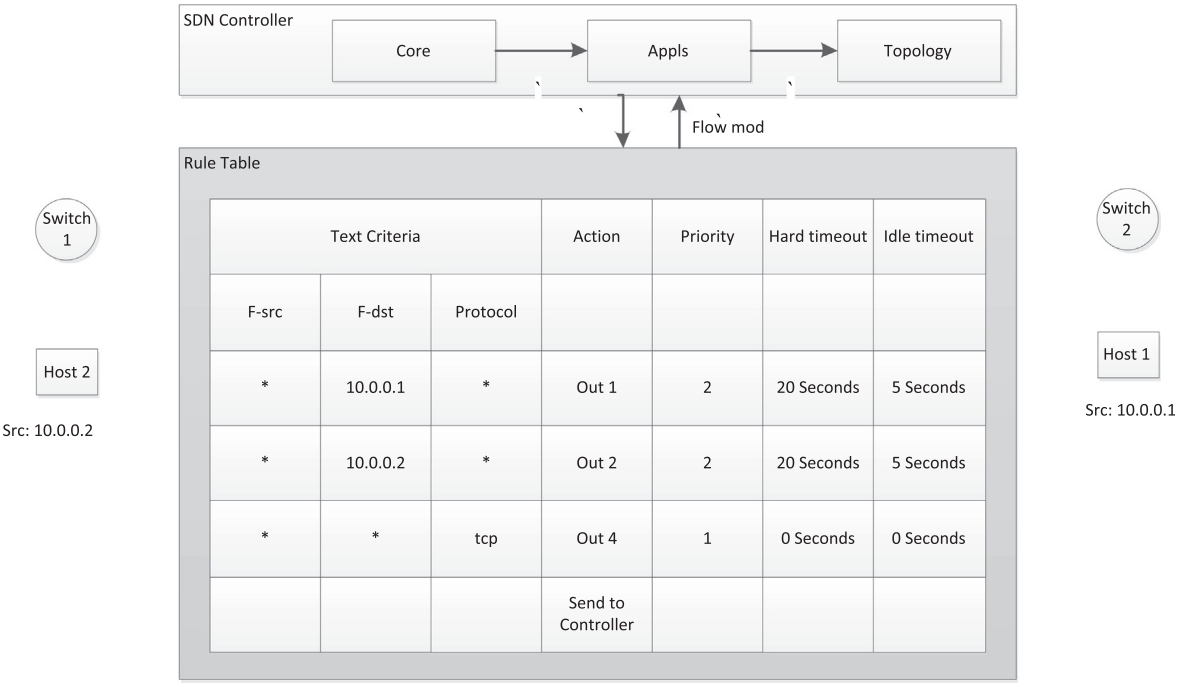
The rule table has rule information and the controller gets these arrangements from the application layer [13,14]. Whenever the packet arrives at switch and switch check the rule from the controller and the controller is not able to find the rule due to the unavailability of its application layer or due to other reasons. The packet waits and causes a delay in the network. Rule arrangement at the application side is called static principle forming and rule forming at the controller side basis on switch demand is called dynamic rule shaping which may be tedious and troublesome [15,16]. To address this issue, a reinforcement learning-based method is proposed that maintains a strategic distance from the bottleneck and provides more efficient rule management at the controller side. This is done by making the automated rule generating process where the controller has the option to build rules dependent on past encounters. The main contributions of this work are as follows:

- Proposed a scenario by which the controller has the option to build rules based on past encounters by using reinforcement learning.
- The introduction of a computerized rule arrangement that incorporates the traffic, stream tables and planning information.

The remaining of this paper is organized as follows: Section 2 presents the software-defined network details and their relation with machine learning-based rule generation. Section 3 presents the related work and problem formation. Section 4 presents the proposed Rule Formation Agent design and development steps. Section 5 presents the experiment results. The Paper concludes in the end with a future direction.

2. Background and related work

The application layer is the top layer of the SDN network which provides services to the controller. The application layer forms the flow rules for the packets in the network. These rules are the basis of packet scheduling. With all of the information related to the network applications and topology of the network, the application layer keeps a check on the state of the network. The application layer communicates with the control layer using the control message. Control Layer, Controller, Control Plane is the central part of the SDN which is also called the brain of the network. The controller is connected to the application layer, and the infrastructure



Text Criteria			Action	Priority	Hard timeout	Idle timeout
F-src	F-dst	Protocol				
*	10.0.0.1	*	Out 1	2	20 Seconds	5 Seconds
*	10.0.0.2	*	Out 2	2	20 Seconds	5 Seconds
*	*	tcp	Out 4	1	0 Seconds	0 Seconds
			Send to Controller			

Fig. 1. An example scenario of flow table.

layer via a protocol called OpenFlow protocol. The controller is connected to the application layer via Northbound API. The controller works as a medium among both layers. The controller receives flow rules from the application layer and saves them into a database when a request for the rule is received and then the controller provides rules for the packet and manages the packet flow of the network. The Data plane of the SDN includes routers and switches which might be physical or virtual, these switches are responsible for the flow of the traffic inside the network. Rules from the controller are sent to the switches which include flow information and all this communication among the control plane and data plane is done via southbound APIs. So switch work according to the rules provided by the upper layer to proceed with the network flow. OpenFlow protocol is also known as communication protocol inside an SDN network. This protocol is used at both ends of the controller, using this protocol Controller interacts with other layers and performs actions in the network. Whenever the switch sends a request for the connection it uses OF to send a request - OF runs on TCP. Fig. 2 shows the SDN architecture and its components.

Flow Tables: Within a switch, there is a flow table that defines what actions should be applied to the packets in an SDN by specifying the match criteria and actions. Match criteria include IP source/destination and protocol or maybe other headers as well. Flow table actions include dropping a packet or sending it to a particular controller. Whenever a packet arrives packet entries are tested with the entries of the corresponding rule table to check which rule goes for which packet. When all the entries are matched, a rule is applied to that packet if not then rejected and looked into the rules table to find the matching rule. If a matching rule is not found controller sends a control message to the application layer to get the rules for the specific packet. The same process initiates when any new entry is found with a packet in the flow table. The structure of the flow table shows in Fig. 3.

The data plane of the SDN include routers and switches which might be physical or virtual, these switches are responsible for the flow of the traffic inside the network. Rules from the controller are sent to the switches which include flow information and all this communication among the control plane and data plane is done via southbound APIs. So switch work according to the rules provided by the upper layer to proceed with the network flow [17]. The following research discusses the previous work in the context of rule formation.

The authors in [18] have worked on dynamic flow rules in SDN. This research focuses on the flow rules dynamically this research proposed a methodology to increase the flow of the SDN by creating own flow entries to keep the network data work efficiently using this strategy switch will be able to form rules locally which means modification needs in the rules will be done on the switch level locally which will increase the flow of traffic in the network. This research presented three different procedures to implement this strategy. And the author claimed that this strategy increases the efficiency of the SDN network. This works as the extended version of the OpenFlow.

Similarly, in another work [19], the authors used Priority-based Flow Control for Dynamic and Flow Management. This research focuses on the issue regarding packet loss in the SDN network which causes discontinuation in the network flow in the data layer. The strategy provided in this research is the change of flow rules based on priority in case of the problem. Traffic misfortune can happen in current SDN equipment switches when the sending rule being applied to a presently dynamic traffic stream is adjusted during continuous traffic transmission. This is credited to the handling dormancy, which is the time expected to alter the sending rule in an equipment switch. This preparing idleness can cause transmission interruption, which additionally prompts bundle misfortune for a transient timeframe. This sort of traffic misfortune can bring extensive execution problem in an SDN arrange, since the problem of packet loss not only causes issues in the disintegration of the system execution yet, also, triggers the transmission issues in the system which thus antagonistically impacts the start to finish transmission execution at the transport layer or the application layer of the system. The mechanism provided by the author is Priority-based flow control which controls the delay in the traffic

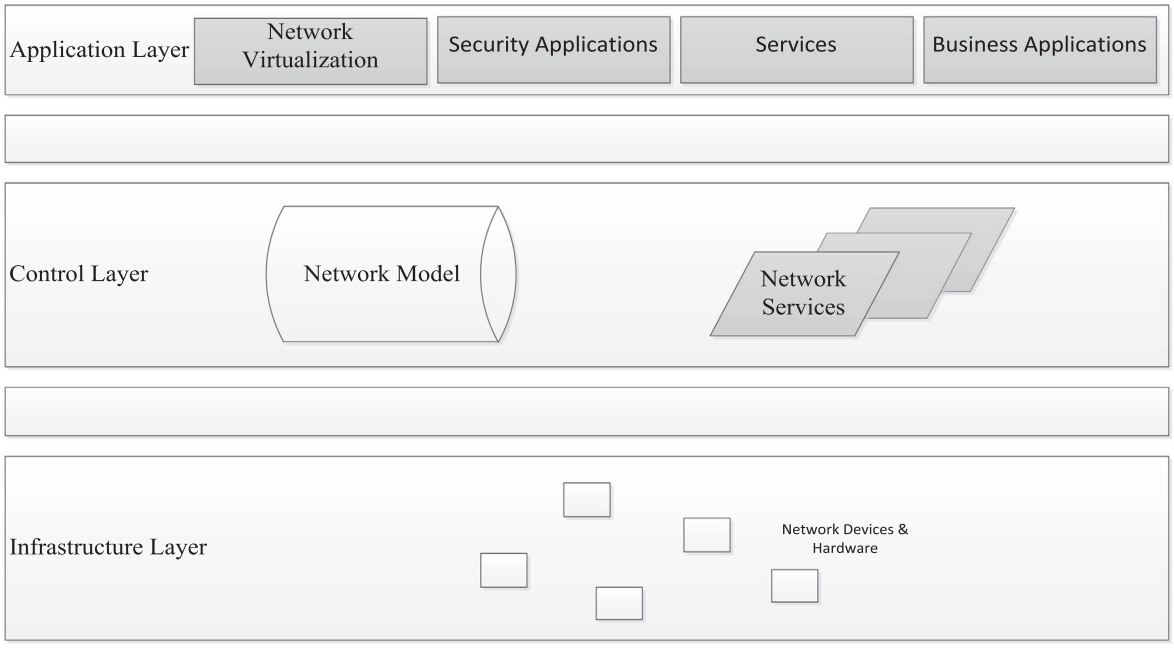


Fig. 2. SDN architecture.

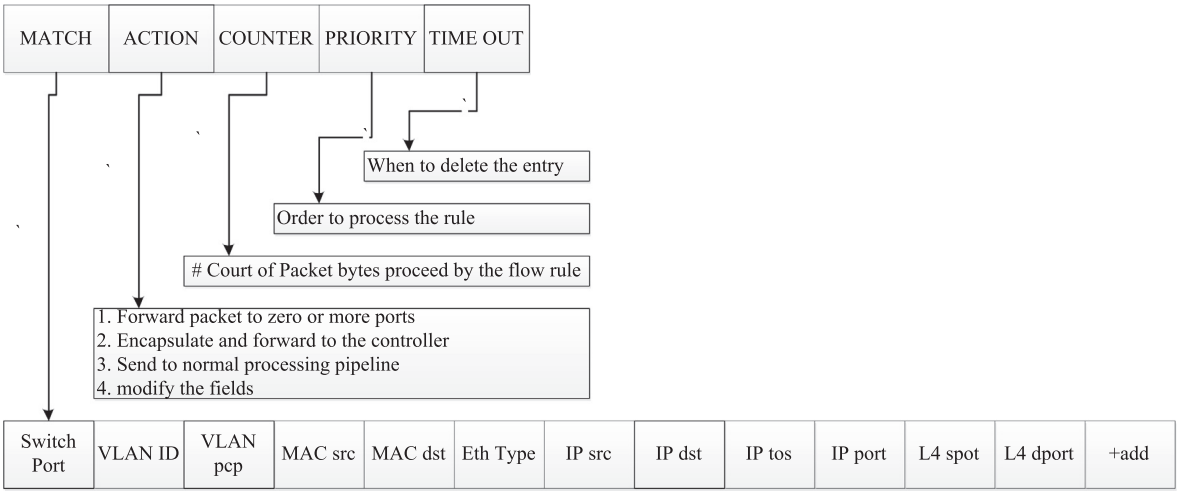


Fig. 3. Structure of flow table.

flow which may save the SDN network from latency by applying this mechanism. This mechanism works as a change of the priority flow of the packet.

Traffic Engineering in SDN has been explored by the authors [20]. This research focuses on traffic engineering concerning software-defined networking that how SDN has made traffic engineering more productive this focuses on the measurement and management of the network flow inside a software-defined environment this research presented a scheme to manage traffic in the network which consists of these two factors include management and measurement and provide parameters which include traffic in the network and topology used by the network and count of the upcoming and going traffic the protocol which is used by the controller to detect the topology of the network is called Link Layer Discovery Protocol (LLDP), this strategy will be able to measure the flow of the network so the network will be ready to cope up with that flow by keeping in check the details of traffic engineering it will be easy for a software-defined network to allocate resources accordingly plus scheduling of the traffic will become easy.

For flow entry management, reinforcement learning has been adopted by authors [21]. This research is related to the problem of flow rule missing this research focuses on when a rule is missing for a particular packet a request is sent to the controller for the missing rule and the controller send a control message to the application layer in the request of missing rule author claims that this process is the time taken and cause a delay in the processing. The author uses reinforcement learning machine learning techniques to solve this issue. The author claims that the techniques he uses gave 60% accuracy when implemented. The author referred to the procedure in which messages are circulated from switch to application layer as Network Control Overhead (NCO). The presented solution works as an alternative provider to keep the flow going so there will be fewer delays the only shortcoming of this research is that it is not yet applied in the field. Emulation is done using machine learning.

An Intelligent Rule Management Scheme for Software Defined Networking has been designed in which the authors focus on rule management in the Software-defined network [22]. This research follows the perspective that when flow rules are transmitted from the application layer to switch network mass network resources are used in this process so this research proposed a methodology to make rule management system easier and error-free without the use of extra network resources. For this purpose, the technique provided by the author is hybrid. This scheme works as compensation between the performance of the network and also related to the flexibility of the system this scheme divided rules into two categories and focuses on the pre-installation of rules on the network switches using the algorithm based on artificial intelligence to deal with the issue of the latency and mass resource consumption in the software-defined network author shows simulation-based results of the used scheme which shows that if this scheme is applied in the field it can improve the network traffic by managing the flow rules in a better way.

In this research [23], a structure is provided to make the flow management system work properly without hurdles. Each flow rule consists of source id destination id and time out after specific time flow rule expires or the time which a packet needs to be processed this process requires time this research uses filters to find out the clean-up time after the processing of rules in the switch. This research is based on minimizing the communication between the SDN controller and the switch placed in the data layer by using techniques to make this process work automatically in case of need this structure also remove the unnecessary or unused rules from the switch to bring low the load on the switch in the network plus this mechanism will keep the important flows for the longer time in the memory of the switch by increasing their time.

A lot of work has been done in predicting the traffic patterns and accordingly installing them on the required switch to reduce the installation of large flow entries, which in itself is cumbersome and leads to inefficient utilization of the network resources. Similarly, the unavailability of rules to be installed on the switch by the controller is also a crucial issue. Rules forming is sometimes a critical and time-consuming process that increases the latency ratio by creating a bottleneck at switch level due to missing rules so to avoid bottleneck due to rule absence, and reduce latency we need to find a way for the controller to form rules without the involvement of application layer.

3. Proposed work

The use of machine learning in solving networking problems has been escalated and has given astonishing results. In the domain of SDN, the use of reinforcement learning has also been witnessed especially for the management of flow rules. In this research a Rule Formation Agent (RFA) is proposed that act as a sub-module in the classical SDN controller, this works on the mechanism of reinforcement learning where a database is maintained for the flow rules that are obtained from the application

layer and in turn these flows are contained with packet id, source, and destination. Every switch at the data plane is provided a flow table which includes flow rules for each packet that arrives at its interface. Therefore, whenever a packet arrives at a switch, it provides a rule concerning the packet and if a rule is missing an open flow request is sent to the controller in response to which controller sends a control message to the application layer which in response provide a missing rule for the packet at the data layer. But if the application layer did not provide the rule, due to any reason, this proposed approach builds rules for the packets applying reinforcement learning on the previously available rules in its storage.

Reinforcement learning has been used that works with the rule of reward and algorithm train the model with an agent. The agent gets a reward and penalty based on its responses to the learning algorithm. The agent learns from the experiences of the system to get the continuing award. This rule formation agent works as a learning agent in the environment of software-defined networking. Which monitors the traffic of the network as well as the rules already saved in the database. Every time a packet arrives at the switch this agent monitors the behaviour and responds when the threshold is achieved and as a result gets the reward of forming the rule for the packet. This behaviour indicates how the agent is working in the software-defined networking environment and how well the model is training to fulfil the job assigned to form the rules. If the threshold value is not met then the system does not respond, and the normal flow of the packet goes on. Agent in the system is expected to respond as required by the system to gain the desired output out of the model. The optimal behaviour of the agent can be represented as π. The space of the state of the agent can be represented as p and the actions performed by the agent in the system can be represented as q and the equation of the system for the ultimate reward will be $q(\pi : p \rightarrow q)$. This module will relate to the database of rules inside the controller which relates to the application layer and get rules from the application layer which include source, destination, id, etc. these combined rules form the flow rule. These rules are different for each packet on the network whenever there is an issue of missing rule or the rule provided does not match with the packet or a rule is not found for the particular packet this module will generate a rule for that packet based on the rules saved in the database inside the SDN controller. This model is presented in Fig. 4.

3.1. Structure of RFA

This section briefly describes the structure of the proposed model. This model works as an agent in the controller. In the structure of software-defined networking communication medium is the controller and all communication revolves around the controller to understand the architecture of this model first we need to revise the working environment of the software-defined network in the software-defined network we are given 3 layers one works as the administrator which provide the services to other layers and 2nd layer works as the communication medium and the backbone of the network whether the third layer of the network deals with the network so three of them are interconnected. So whenever a packet arrives at the data layer rule tables are given to the switch, switch process the packet according to the specific rules provided for the specific packet whenever a rule is missing in the table switch sends a request which is popularly known as the open flow request in response to this request controller sends a control message to the application layer now when the message received at the application layer provides the rule for the specific packet and traffic goes on, this is the basic flow of the software-defined network.

To understand the proposed solution, consider the following scenario: A packet arrives at the data layer and in the rule table switch cannot find a rule for that particular packet switch sends an open flow request to the controller in response to which controller sends a control message to application layer which is under

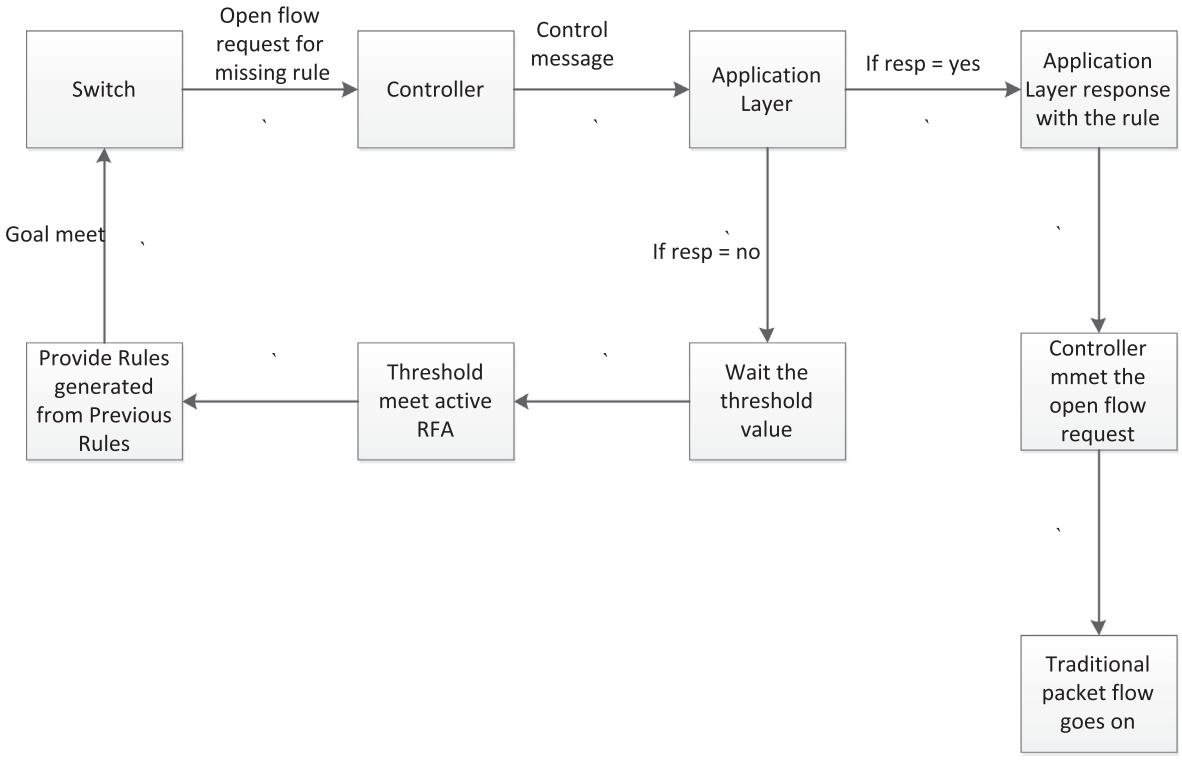


Fig. 4. Proposed workflow presentation of SDN.

attack or cannot provide a rule for the particular packet how the network is going to proceed? Consequences include Packet loss, delay, and latency, the bottleneck at the switch. The following sub-section shows the rule formation agent.

3.2. Working of RFA

Whenever a packet arrives at the switch a rule table is provided at the switch to check the corresponding rules for each packet. Scenario: Rule is not provided for a packet in this case as switch contains a repository which keeps the stats for each packet if a packet has no rule assigned by the rule table then switch will check for the table hit ratio if that ratio is greater than the threshold than request will proceed to the RFA and agent will provide a rule for that packet as in controller rules are saved in a database RFA is connected to that database and using previous rules form rules for the new packet and network traffic proceeds. And if the threshold value is less than the value set by the module then as a penalty no new destination is assigned. This system works with reinforcement learning. Whenever a packet arrives and the rule is not assigned by RFA this packet is sent to the port which is available, and the value is greater than the threshold. This module works with the source and destination of the packets whenever the destination is not assigned every port available in the network is assigned as the destination for that packet to keep the traffic flow going in the network. Whenever the threshold point is reached RFA is assigned with a reward of providing the rule for the packet. The complete architecture of the SDN controller after a proposed model is given below. The placement of RFA in the controller is shown in Fig. 5. The notations used in the algorithm for RFA is presented in Table 1 and the algorithm is represented in Table 2.

4. Experiments and results

In order to evaluate the performance of the proposed RFA, the simulation is performed. The tools used for the implementation of this research are open daylight controller and Mininet and the platform used is Ubuntu which was installed on a virtual environment using a virtual box. For benchmarking with classical SDN implementation the throughput and packet loss and are considered. Besides, an important metric flow insertion delay is also considered that reduces the overall delay for the rule insertion which

Table 1
Notations and their meanings.

Notations	Meanings
Tcp	Transmission control protocol
rfa	Rule formation agent
C	*Controller*
μ	*Threshold*
S	*Switch*
Cm	*Control message*
Al	*Application layer*
Dl	Data layer
Of	*OpenFlow request*
fr	*Flow rule*
rt	*Rule table*
l	*Latency*
r	*Response*
p	*Packet*
Ps	*Packet submitted*

Table 2
RSA algorithm.

1. Input: S (p, rt) Output: ps (Packet submitted)	
1:	2. S->Of
2:	3. Of-> C
3:	4. C*Cm
4:	5. If cm = yes
5:	t%<μ -> r = ps
6:	6. else if cm = no
7:	d->∞ (dead end)
8:	while cm = no
9:	r = ps
10:	t%>μ
11:	for all t%>μ do
12:	rfa->r
13:	rfa(r + rg)
14:	r + rg = t%>μ
15:	s->r
16:	if r = r then
17:	r = ps
18:	7. end if
19:	8. end for
20:	9. end while
21:	end if

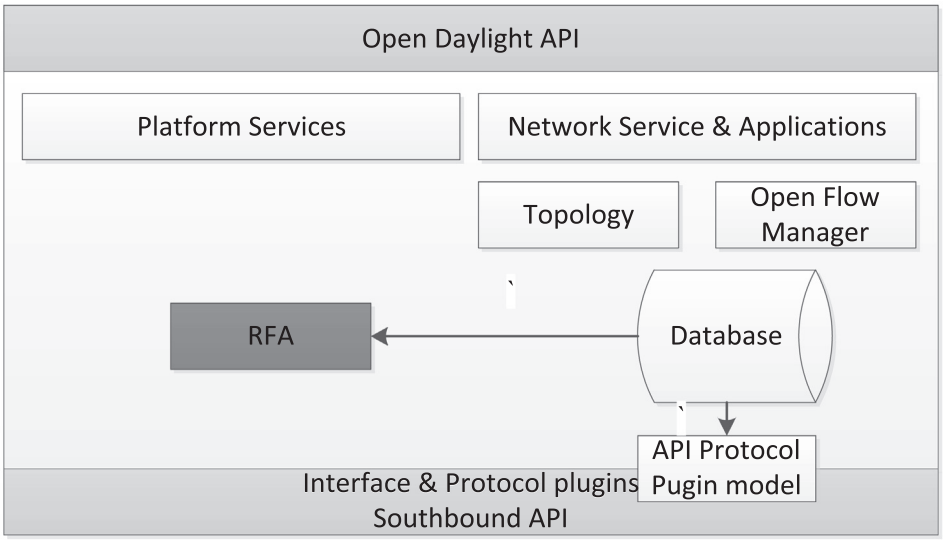


Fig. 5. RFA placement in SDN controller.

includes the reduction of the application layer involvement and waiting time for its response. In other words, the insertion delay time comparison provides the information on how long it takes for the classical SDN to respond to the request for the rule and how long the proposed model took to respond to the request regarding the missing rule. These experiments are done on the basis of parameters defined in Table 3.

4.1. Packet loss ratio

Whenever a flow rule is missing, predictably, packet loss happened because more time is taken by the network to provide the rule. After all, more latency which as a result aids the packet loss that happens due to bottleneck at the switch. When a lot of packets are in the queue and rule is missing for a packet it not only go towards the packet loss of the packet with the missing rule problem but can also aid the packet loss of other packets reside in the queue.

The proposed RFA improves the efficiency of the SDN and decrease the ratio of the packet losses as shown in Fig. 6. This is achieved by providing rules for the packet which as a result does not cause bottleneck or congestion at the switch and as a whole increases the overall performance of the software-defined network.

4.2. Flow rule insertion delay

Fig. 7 shows the correlation between the system response under different conditions how much time a packet takes usually to process in the network and how much time taken by the packet whose rule is missing or not found and the system solve the problem traditionally so how much time is taken by the system to solve this problem manually how much time is taken by the switch to send an open flow message to the controller and how much time taken by the controller to communicate with the application layer in order to solve the particular issue regarding missing rule so this time and latency periods is compared with the time taken by the proposed model to activate and respond to the request when the threshold value is meet so model activates and provide a rule for the packet to make sure that the network traffic won't get affected by this issue. This graph shows how the insertion delay decreases with the use of our proposed model in order to deal with the missing rule issue.

Table 3
Experiment parameters.

Parameters	Values
Network Size	1000*1000
No. f Controllers	1
No. of Switches	20
No. of Hosts	35
Emulation Time	2000 s
Packet Size	512 bytes

4.3. Throughput

Fig. 8 shows the comparison of the throughput of the proposed system with the throughput of the traditional system the throughput of the system increases as the proposed solution provide solutions for the problems of the traditional network so this shows that whenever a packet gets facilitated with the rule provided by the RFA it increases the throughput of the overall system as it helps to remove the flaw of the system by providing a solution.

4.4. Discussion

This section includes the discussion about the results shown above in the graphs. These graphs show the comparison of the system, its throughput, and how the system responds to the mechanism proposed. Fig. 8 shows that when the throughput of the system increases then how the system reacts to the problem traditionally and how the system responds when the model starts working. It is clearly shown that the throughput of the system increases which indicates that the model is working fine in the system although each research and technology needs improvement over time. The same will happen in our case but as an effort the results show that this problem of the missing rules can also be solved this way. The basic purpose behind this research was to low the impact of the problem on the software-defined networking environment. The results show that in the case of dos attack on the application layer or due to any other reason if the application layer is not connected to the controller and cannot provide rules for the packets the proposed solution can tackle by providing solutions. As of now, there will be less amount of latency and packet loss due to the rule absence so it is the little contribution of our research that can be implemented in the field also. In comparison to the previous research, there is a lot of research going on in the field of software-

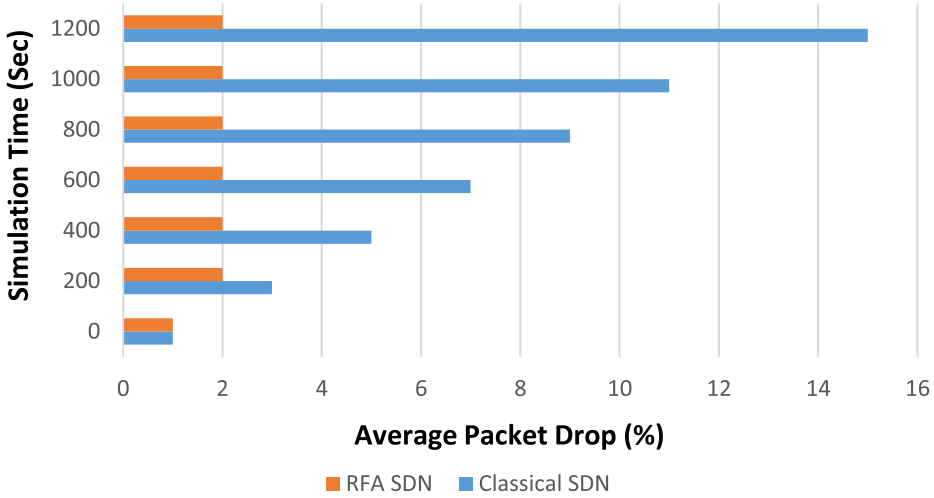


Fig. 6. Comparison graph of packet drop ratio.

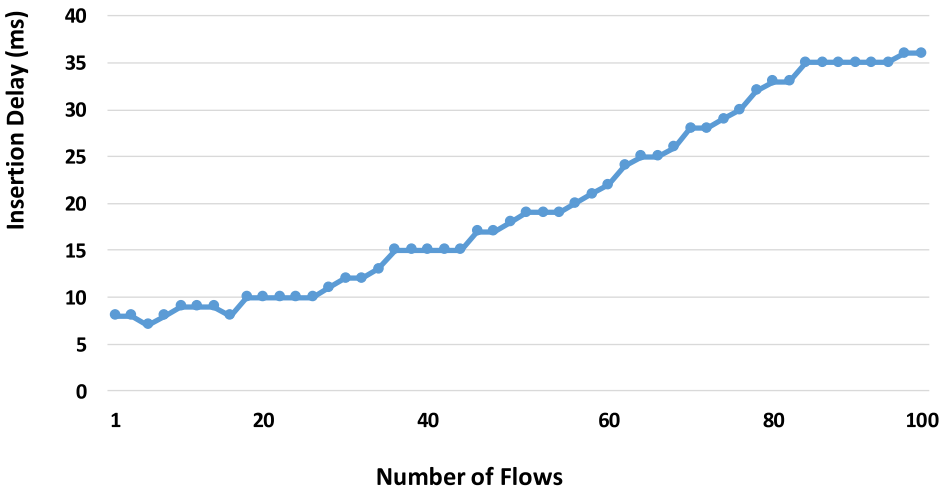


Fig. 7. Comparison of delay in the time of rule insertion.

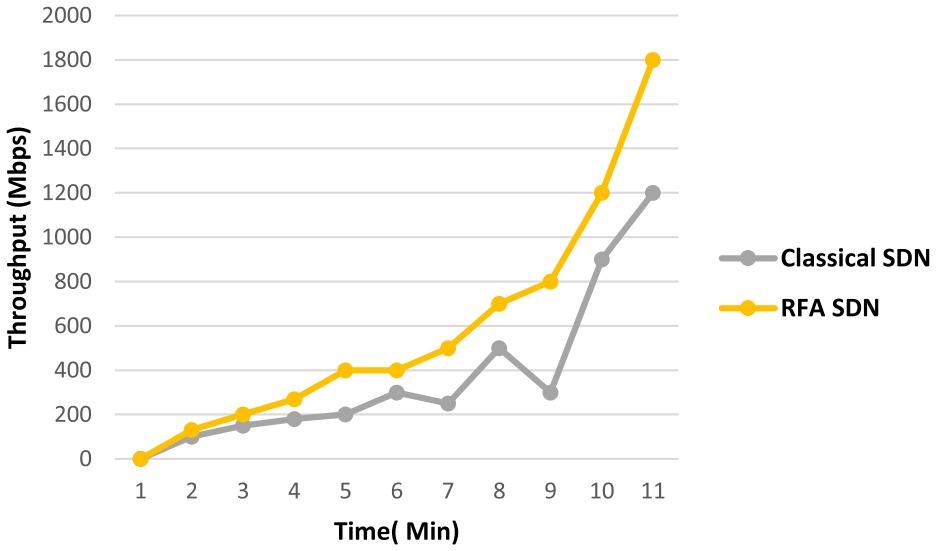


Fig. 8. Throughput comparison.

defined networking as this networking has not yet reached its peak so a lot of areas for research are still open in the field. The previous research do not own such a mechanism that make rules by the controller itself. However, some researches need improvement to get used and implemented in the field our research is in term of that scenario presents a different approach as in one criterion we are trying to solve multiple issues that can help fight the issues related to latency and packet loss. Additionally, reduce the risk of deadlock at the end of the switch due to missing rules and this research can be used as part of an artificial intelligent controller.

Machine learning techniques can be used to form rule formation automated as witnessed in this work that how we used reinforcement learning to form rules by adding a module in the SDN controller and used reinforcement learning to train that module as reinforcement learning work with the rule of reward and penalty so we used that rule to train our model which was later put in a scenario where It has to form rules automatically upon the request of the SDN switch. As when application layer provides rules for the packets those rules got saved at the controller temporarily and we connected our module with that temporary storage on the basis of which we train our module and those past rules helped the module to do its job automatically in order to deal with the problem at the switch. We used default rules sent by the application layer in order to train our model which turned out to be a good approach in order to train the model. As we discussed earlier that reinforcement learning works on the basis of reward so it became easy to train a model using machine learning plus circumstances on the SDN controller switch indicates a way of using the reinforcement machine learning approach in order to train our model and make our model work well to reduce the effect of the problem in the software-defined environment. So we answered all the questions we were supposed to answer in our research.

5. Conclusion

Software-defined networking is a networking approach that separated the control plane from the data plane. This approach provides programmability and attracted many researchers and companies by the time. It invented in a very short time this

approach gained huge popularity with each positive aspect of technology. We have to deal with the drawbacks also in terms of software-defined networking; there are loopholes which need to be solved as in our research. We followed a problem at a traditional software-defined network which is regarding the flow rules as flow rules plays an important role in a network as they keep the network traffic going so problem regarding them leads the whole network to a deadlock situation. The other aspect when a security threat hits the system in that case too our module finds a way to keep the network traffic going whether a system is under attack or there is any other issue regarding missing rules. We proposed a methodology that works with the mechanism of Reinforcement learning to form flow rules automatically and our results show that this approach can help to improve the software-defined networking environment. Machine learning algorithms have been used in a lot of researches and regarding solving many areas of problem in software-defined networking so our research can become part of a big intelligent software-defined networking environment. Latency has been a considerable problem of the SDN environment which is taken care of in our research plus the issue regarding missing rule is also taken care of and the main scenario which was the base of the research that if at some risk application layer got disconnected, how the traffic is going to process in the network. In future, other supervised learning techniques will be used to perform auto rule formation.

Declaration of Competing Interest

The authors declare that there is no conflict of interest regarding the publication of this article.

Acknowledgement

This work is partly supported with the financial support of the Science Foundation Ireland grant 13/RC/2094_P2 and partly funded from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 754489.

References

[1] Macedo DF, Guedes D, Vieira LF, Vieira MA, Nogueira M. Programmable networks—From software-defined radio to software-defined networking. IEEE Commun Surv Tutorials 2015;17(2):1102–25.
[2] Cao X, Yoshikane N, Popescu I, Tsuritani T, Morita I. Software-defined optical networks and network abstraction with functional service design. J Opt Commun Network 2017;9(4):C65–75.
[3] Hu Q. Dynamic policy based software defined network mechanism. ed: Google Patents 2020.
[4] Braun W, Menth M. Software-defined networking using OpenFlow: Protocols, applications and architectural design choices. Future Internet 2014;6 (2):302–36.
[5] Iqbal S, Qureshi KN, Shoaib F, Ahmad A, Jeon G. Minimize the delays in software defined network switch controller communication. Concurr Comput Pract Exp 2020:e5940.
[6] T. Kohler, On consistency and distribution in software-defined networking, 2020.
[7] Isyaku B, Mohd Zahid MS, Bte Kamat M, Abu Bakar K, Ghaleb FA. Software defined networking flow table management of openflow switches performance and security challenges: a survey. Future Internet 2020;12 (9):147.
[8] Truong Dinh K, Kukliński S, Osiński T, Wytrębowicz J. Heuristic traffic engineering for SDN. J Inf Telecommun 2020;4(3):251–66.
[9] Iqbal S, Abdullah AH, Ahsan F, Qureshi KN. Critical link identification and prioritization using Bayesian theorem for dynamic channel assignment in wireless mesh networks. Wireless Netw 2018;24(7):2685–97.
[10] Hu J, Reed M, Al-Naday M, Thomos N. Blockchain-aided flow insertion and verification in software defined networks. In: 2020 Global Internet of Things Summit (GIoTS). IEEE; 2020. p. 1–6.
[11] Böhm M, Jeż Ł, Sgall J, Veselý P. On packet scheduling with adversarial jamming and speedup. Ann Oper Res 2021;298(1-2):7–42.
[12] Su J, Xu R, Yu S, Wang B, Wang J, Redundant rule detection for software-defined networking, *KSII Transactions on Internet & Information Systems*, vol. 14, no. 6, 2020.
[13] Agarwal S, Kodialam M, Lakshman T. Traffic engineering in software defined networks. In: 2013 Proceedings IEEE INFOCOM. IEEE; 2013. p. 2211–9.
[14] Iqbal S, Abdullah AH, Qureshi KN. An adaptive interference-aware and traffic-aware channel assignment strategy for backhaul networks. Concurr Comput Pract Exp 2020;32(11):e5650.
[15] Amaral P, Pinto PF, Bernardo L, Mazandarani A. Application aware SDN architecture using semi-supervised traffic classification. In: 2018 IEEE conference on network function virtualization and software defined networks (NFV-SDN). IEEE; 2018. p. 1–6.
[16] Iqbal S, Qureshi KN, Kanwal N, Jeon G. Collaborative energy efficient zone-based routing protocol for multihop Internet of Things. Trans Emerg Telecommun Technol 2020:e3885.
[17] Tomovic S, Lekic N, Radusinovic I, Gardasevic G. A new approach to dynamic routing in SDN networks. In: 2016 18th Mediterranean electrotechnical conference (MELECON). IEEE; 2016. p. 1–6.
[18] Wei Q, Perez-Caparros D, Hecker A. Dynamic flow rules in software defined networks. In: 2016 Fifth European Workshop on Software-Defined Networks (EWSDN). IEEE; 2016. p. 25–30.
[19] Oh B-H, Vural S, Wang N, Tafazolli R. Priority-based flow control for dynamic and reliable flow management in SDN. IEEE Trans Netw Serv Manage 2018;15 (4):1720–32.
[20] Shu Z, Wan J, Lin J, Wang S, Li D, Rho S, et al. Traffic engineering in software-defined networking: Measurement and management. IEEE Access 2016;4:3246–56.
[21] Mu T-Y, Al-Fuqaha A, Shuaib K, Sallabi FM, Qadir J. SDN flow entry management using reinforcement learning. ACM Trans Auton Adapt Syst (TAAS) 2018;13(2):1–23.
[22] Wang L, Li Q, Sinnott R, Jiang Y, Wu J. An intelligent rule management scheme for Software Defined Networking. Comput Netw 2018;144:77–88.
[23] Challa R, Lee Y, Choo H. Intelligent eviction strategy for efficient flow table management in openflow switches. In: 2016 IEEE NetSoft Conference and Workshops (NetSoft). IEEE; 2016. p. 312–8.