A New Machine Learning-based Collaborative DDoS Mitigation Mechanism in Software-Defined Network

Saif Saad Mohammed*, Rasheed Hussain*, Oleg Senko*, Bagdat Bimaganbetov*, JooYoung Lee*, Fatima Hussain†, Chaker Abdelaziz Kerrache‡, Ezedin Barka§, and Md Zakirul Alam Bhuiyan¶
* Innopolis University, Innopolis, Russia
Email: {o.senko,r.hussain,s.s.mohammed,b.bimaganbetov,j.lee}@innopolis.ru
† Ryerson University, Canada, Email: fatima.hussain@ryerson.ca
‡ University of Gharadaia, Algeria, Email: kerrache.chaker@univ-ghardaia.dz
§ UAE University, UAE, Email: ebarka@uaeu.ac.ae
¶ Fordham University, USA, Email: zakirulalam@gmail.com

***Abstract*—Software Defined Network (SDN) is a revolutionary idea to realize software-driven network with the separation of control and data planes. In essence, SDN addresses the problems faced by the traditional network architecture; however, it may as well expose the network to new attacks. Among other attacks, distributed denial of service (DDoS) attacks are hard to contain in such software-based networks. Existing DDoS mitigation techniques either lack in performance or jeopardize the accuracy of the attack detection. To fill the voids, we propose in this paper a machine learning-based DDoS mitigation technique for SDN. First, we create a model for DDoS detection in SDN using NSL-KDD dataset and then after training the model on this dataset, we use real DDoS attacks to assess our proposed model. Obtained results show that the proposed technique equates favorably to the current techniques with increased performance and accuracy.**

***Index Terms*—DDoS attacks, SDN, Machine Learning, Security**

I. Introduction

The current communication networks are based on TCP/IP communication architecture. This IP-based architecture on one hand has driven the course of the entire communication networks, but on the other hand faces several pressing issues. Inefficient network management, complex routing and operational scenarios, security loopholes, vague distinction between control and data channels, are to name few [1]. Due to recent advancements in communication technologies and the birth of Cyber-Physical System (CPS), every application has moved to Internet. Therefore, this traditional architecture has to handle new challenges such as mobility, cross-layer functionality, scalability, security, and privacy. However, it was never designed to handle distributed real time applications incorporated with context awareness and runtime scalability requirements. To date, several temporary solutions have been put forth, such as layered security, routing, IPv6-based addressing, to name a few. These solutions may work in the short run, but cannot cope well with the increasing demand of Internet-hungry real time applications. One possible solution is to manage the data and control separately. In this context, Software-Defined Network (SDN) has paved its way to offer the decoupling between data and control of the network.

A. Software-Defined Networking (SDN)

SDN has emerged as one of the promising solutions to cope with afore-mentioned challenges. Some of the SDN's noteworthy features include separation of control and data plane, network control through software, efficient resource utilization, flow abstraction from application with agility, and programmable networks. SDN offers numerous benefits including on-demand provisioning, load balancing, and scalability in network resources with data needs. Furthermore, SDN also helps administrators manage the entire network consistently and holistically, regardless of the underlying network technology.

SDN separates the data plane and control plane, in contrast to traditional networks in which same plane is shared between data and control traffic. SDN architecture is comprised of three blocks; SDN controller (which is the implementation of the control plane), Open-Flow protocol (communication protocol between SDN controller and network device), and the SDN device (a router or a switch).

Although SDN provides numerous benefits over the traditional network architecture, it also poses new challenges to the network security. SDN controller being single point of failure, it is prone to distributed denial of service (DDoS) attacks. DDoS is notorious of all security threats that challenge Internet, therefore it is essential to mitigate its effects in SDN [2]. Traditional ways to face DDoS attack include, but not limited to, adding more capacity, blocking source-based traffic, and behavioral analysis. However, these approaches have their own limitations in the face of massive DDoS attacks. Therefore, careful formulation of rules and policies followed by its implementation at SDN controller is required for SDN security. It is important to note that complete network knowledge and real expertise are required to define the set of rules on the SDN controller to block malicious activities and intrusions into the system. Machine learning (ML) techniques can be effectively applied to formulate such rules and policies and can untap the real potential of SDN. Furthermore, new ML-based security solutions can be designed and then benchmark results are used to prove the accuracy of proposed model.

B. *Machine Learning and DDoS*

ML has emerged as an efficient technique for providing system security; such as authentication, access control, anti-jamming offloading and malware detections [3], [4]. Support Vector Machines (SVM), naive Bayes, K-Nearest Neighbor (K-NN), and neural network/Deep Neural Network (DNN) are used to label and classify the network traffic, for DDoS attack. In the absence of labeled data, multivariate correlation analysis and Q-learning reinforcement learning (RL) can be used to detect DoS attacks. Due to advancement in technology and the shift from traditional Internet architecture to SDN, ML has been used in network and routing functions, as well as for network security [5].

Our main idea is to use ML techniques to mitigate DDoS attacks in SDN. In essence, we design an ML-based system and train it over an existing dataset to optimize the detection and mitigation of DDoS attacks. The intelligence of ML together with the scalability of SDN enable the identification of rouge devices that provide the channel for DDoS attacks. Such devices can be blocked and the DDoS attacks can be mitigated before they cause any damage to the network.

C. *Summary of contributions*

The main contributions of this paper are summarized below: In this paper, we,

- design and dry run a testing environment of multi SDN controllers,
- develop an automated notification system to notify the SDN controller in the attacker's network,
- design and create a machine learning model to detect DDoS attacks,
- and test the accuracy of our proposed DDoS mitigation scheme, using a real-world attack. We combine the benefits of SDN flexibility and ML intelligence and analyze their enhanced efficiency.

The remainder of the paper is organized as follows: in Section II, we outline the related work followed by our proposed scheme in Section III. Experimental results are discussed in Section IV followed by concluding remarks in Section V.

II. Related work

Most of the research carried out in DDoS focuses on the target of an attack, and attempts are made to block the IP of the attacker. This technique becomes inefficient in NAT-enabled SDN networks where SDN switch overflows due to traffic blocked by the SDN controller. This degrades the network performance and there is a need of efficient, viable and cooperative defense mechanism. In this regard, Hameed et al. [9] proposed an SDN-based DDoS mitigation scheme, which allows collaboration among various controllers on different autonomous systems (ASs).

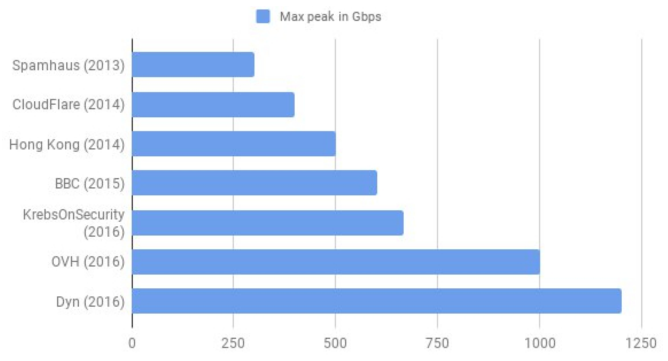

Fig. 1. Biggest DDoS attacks in the past 4 years [6], [7], [8]

In another work, Yang et al. [10] proposed an SDN-based DDoS attack detection framework with cross-plane collaboration. Authors used lightweight flow monitoring algorithm, which utilizes OpenFlow switch counters for filtering abnormal traffic. Conti et al. [2] proposed a sequential analysis-based approach to detect DDoS attacks in SDN. While Zheng et al. [11] proposed a real-time DDoS defense mechanism by using adaptive correlation analysis. This mechanism is used to defend against popular DDoS attacks such as SYN flooding, crossfire, and UDP-based amplification. A packet analysis technique is proposed by Gkountis et al. [12], in which packet analysis was performed to find anomalies in traffic.

In addition to above mentioned techniques, ML has also been utilized by research community, to combat DDoS attacks. Meti et al. [13] used ML techniques such as SVM and neural networks (NN) to classify different connections in the SDN. Similarly, Barki et al. [14] proposed ML-based clustering techniques to detect DDoS attacks in SDN. Authors used various ML algorithms such as K-means, naive Bayes, and K-nearest neighbor to detect anomalies in the traffic. In [15], authors analyzed various ML techniques, such as SVM, fuzzy logic, decision tree, neural networks, and Bayesian networks, to detect DDoS attacks in the network. However, their schemes did not contain DDoS mitigation technique.

To this end, most of the existing works are vulnerable to misbehavior attack, as listed in [5]. The misbehavior attack refers to a situation when the attacker sends a normal traffic to the target, which causes the SDN controller to install an IP-based flow on the switch, after that the attacker launches DDoS. Additionally, in [9], the authors proposed a communication protocol between the controllers in SDN; however, they did not provide any authentication mechanism or a mitigation technique for replay-attack.

From the above discussion, it is imperative to think of an intelligent mechanism with less network overhead to detect and mitigate DDoS attacks in SDN. To fill the gaps, our proposed solution integrates ML intelligence with the programmability power of SDN, along with a reliable communication protocol (among SDN controllers). Furthermore, our proposed solution is not vulnerable to the afore-mentioned attacks due to the following reasons:

- An IP based flow is installed on the switch which will forward the packets from source to destination.
- A matching IP based flow is installed, therefore, packets are sent less frequently to ML server via controller.
- Moreover, communication protocol between controllers provides an IP based authentication module having packet time-stamp which mitigates replay attacks

III. Proposed DDoS Mitigation Scheme

A. Problem Description

The last four years have witnessed a massive scale DDoS attacks on different infrastructures. Figure 1 summarizes some of the well known DDoS attacks.

It can be observed that every new year has 200% increase in DDoS attacks as compared to previous year. Such attacks cause severe disruptions in the services and therefore, massive traffic will cause dire consequences to the companies that have been set as targets. Some reports mention that some financial companies report a loss of $100,000 per hour [1] [2] [3], in addition to the loss of customer trust. Starting from 2016, most of DDoS attacks were caused by compromised IoT devices (used as bots) where such devices consume the Internet bandwidth thereby resulting in deteriorating Internet speed and increasing Internet invoice. In case of IoT, mitigating these attacks from the target side will be very hard, if not impossible due to the large number of compromised and attacking devices. To make things worse, some of these devices could be behind a NAT, therefore the companies will end up blocking a large area of the Internet. Therefore, a collaborative mechanism among the Internet Service Provider (ISPs) and the big companies is needed to mitigate these kinds of attacks.

B. Methodology and Environment

To mitigate DDoS attacks in SDN, we set up a networked environment with 4 SDN controllers, Ryu [16], POX [17], ONOS [18], and OpenDayLight [19] for individual Internet Service Providers (ISPs) connected through Internet as shown in Figure 2. Each controller has it own tool for development, RYU and POX are developed in python whereas ONOS and OpenDayLight are developed in Java.

Every controller can communicate with other controllers through Internet and with the attack detector server (the ML server). In Figure 2, the controller of ISP 1 is the target device and on ISP 2, 3 and 4 we have the attackers who launch attacks on the target device(s).

An initial flood flow with low priority is installed on each controller to prevent any delay that will be caused by the packet analysis. Afterwards, the initial flow forwards any packet to all the ports and also to the controller. This mechanism works for each ISP, so that we will not have to wait until the ML server responds. In other words, when a packet arrives to the switch, it will be forwarded to the controller and trigger the *packet_in* event/function. Then the controller extracts the needed features for the ML server, and thereby sending it to the ML server. When the extracted features are analyzed, the machine learning server responds whether the packet is normal traffic or not. Based on the result, the controller either installs an IP base flow, or it communicates with another controller to create a deny IP based flow.

C. Application

We developed a server application where we implement the logic for attack mitigation. The application was developed with Python and it can be run as standalone server or together with controller application. The server application consists of the following modules:

- **Authorization Module:** This module checks if the controller can send requests to the server or not.
- **ML-based Prediction Module:** The main module in the application that intelligently predicts the potential DDoS attacks.
- **Wrappers:** This module implements different wrappers for Java/Python programming language.

The structure of the application is shown in Figure 3.

It is worth noting that the deployed controllers use two different language (Python and Java), therefore we wrote separate wrappers as a library, to help in communication between controllers and with the ML server. For such communication, we need an import library that implements abstract methods and writes configurations, such as server address, port, etc.

ML server uses sockets for communicating with controllers and uses ML techniques to filter traffic. All communication between server and controller are in JSON format. Figure 4 outlines the detailed overview of the implementation and packet validation.

D. Packet Validation

Prior to packet validation, an authenticated connection must be established between controller and ML server. For authentication, the controller sends a token to the server, and if the token is validated, a connection between server and controller is established, otherwise, the server will close the connection and the controller has to wait for some time before initiating the authentication process again. This prevents the ML server from being flooded with authentication packets. The authentication is optional and can be disabled at server-side. When the controller starts receiving packets, the packet validation process begins immediately. SDN controller will start the features extraction in two cases. i) accumulate n sessions and start processing these sessions, and ii) process the accumulated sessions after a particular time interval (in case the accumulated number of sessions is less than n). By applying these two cases, There will be no delay in case the SDN controller is not under high-load. Also, n can be set dynamically based on the current load and the processing power of the SDN controller. After the features extraction, the SDN controller will send the list of extracted features to ML server. The server returns the chance of packet abnormality

[1]https://www.businesswire.com/multimedia/home/20150420005307/en/

[2]https://www.helpnetsecurity.com/2017/05/02/ddos-attacks-money-risk-per-hour/

[3]https://lp.incapsula.com/rs/incapsulainc/images/eBook%20-%20DDoS%20Impact%20Survey.pdf

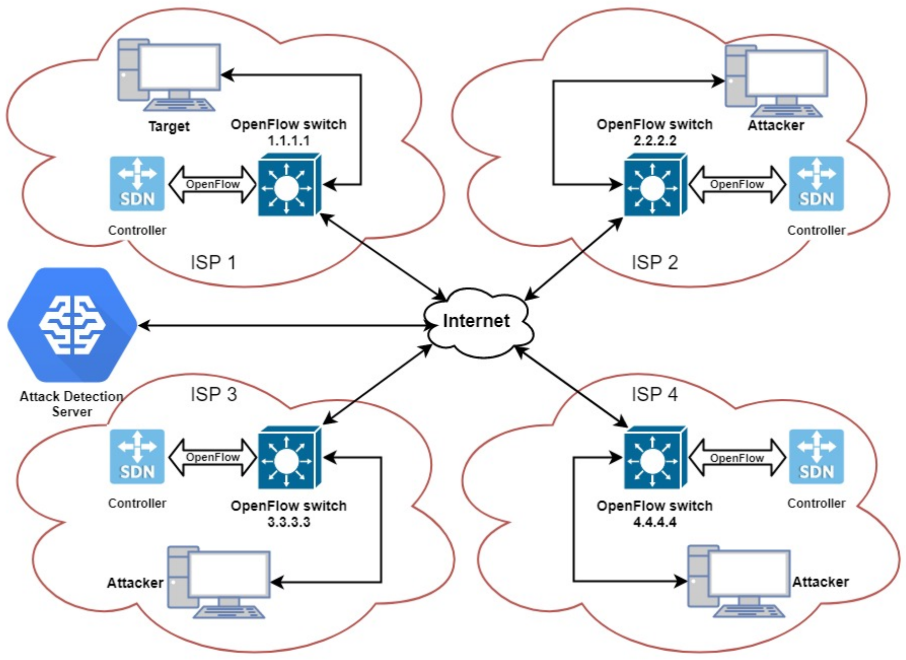

Fig. 2. Network Model

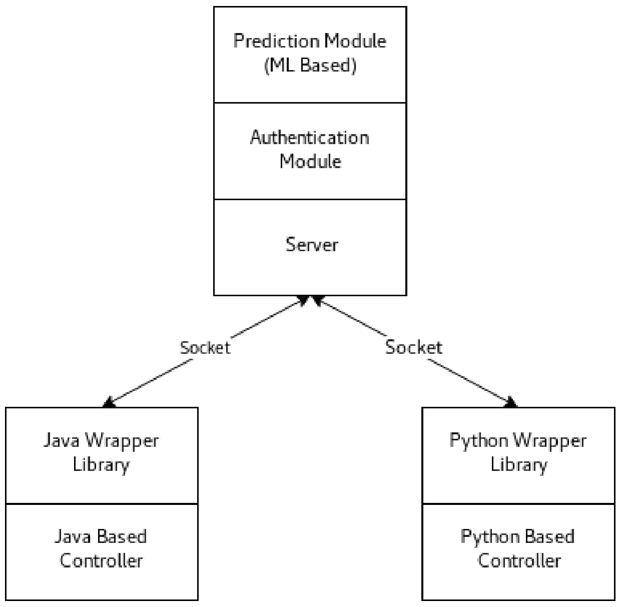

Fig. 3. Structure of the server application

(in percentage) for each packet. Based on the received result, the controller makes a decision. If the packets are normal, the controller creates an IP-based flow with **allow** as an action and applies this flow on the switch. Note that even if IP address is flagged as normal, packets from this IP will still be accumulated and will be sent to ML server for further analysis. This prevent the attacker from utilizing the misbehavior attack.

On the other hand, if packets are recognized as abnormal, the target controller sends a request to the attacker-side controller to block the attacker's IP. The attacker-side controller creates an IP-based flow with **deny** as an action and installs it on the switch. Thus, the attacker will not be able to send requests to the target (the switch drops all requests from attacker's IP). These flows (the allow and deny flows) have both soft and hard timeouts, after expiration of which flows will be removed from the switch. If controller does not see any activity from IP in a given time frame, soft timeout is applied where in case of hard timeout, the IP may still be active and may has traffic.

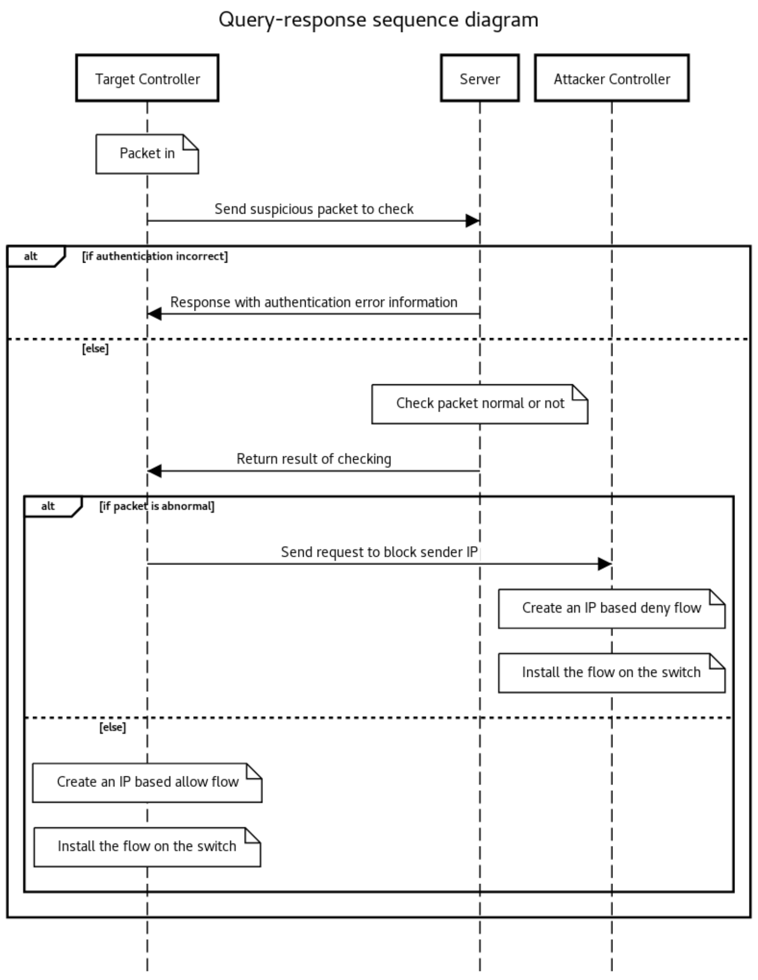
Query-response sequence diagram
Target Controller
Server
Attacker Controller
Packet in
Send suspicious packet to check
alt
[if authentication incorrect]
Response with authentication error information
[else]
Check packet normal or not
Return result of checking
alt
[if packet is abnormal]
Send request to block sender IP
Create an IP based deny flow
Install the flow on the switch
[else]
Create an IP based allow flow
Install the flow on the switch

Fig. 4. Packet validation process

E. Machine Learning (ML)-based Approach

We create a model based on NSL-KDD dataset [20]. This model uses naive Bayes classification detection algorithm. In naive Bayes classifiers, Bayes' theorem is applied with assumption that features are independent of each other. Naive Bayes classifiers are also referred to as probabilistic classifiers. These classifiers have several advantages such as scalability (it scales linearly with the number of predictors and data points), probabilistic predictions, and capability to handle continuous and discrete data.

We created our model and deployed it based on the same features as used in [5] and shown in Table I [21]. The algorithm for selecting the features in order to train the model is taken from [5]. According to this algorithm, three features selecting algorithms (Genetic, Ranker, Greedy) are run together and the results are combined. A total of 25 features were selected from the dataset.

TABLE I
FEATURES IN NSL DATASET

#	Feature Name	#	Feature Name
1	Duration	22	Is_guest_login
2	Protocol Type	23	Count
3	Service	24	Srv_count
4	Flag	25	Serror_rate
5	Src_byte	26	Srv_serror_rate
6	Dst_byte	27	Rerror_rate
7	Land	28	Srv_rerror_rate
8	Wrong_fragment	29	Same_srv_rate
9	Urgent	30	Diff_srv_rate
10	Hot	31	Srv_diff_host_rate
11	Num_failed_logins	32	Dst_host_count
12	Logged_in	33	Dst_host_srv_count
13	Num_compromised	34	Dst_host_same_srv_rate
14	Root_shell	35	Dst_host_diff_srv_rate
15	Su_attempted	36	Dst_host_src_port_rate
16	Num_root	37	Dst_host_diff_host_rate
17	Num_file_creations	38	Dst_host_serror_rate
18	Num_shells	39	Dst_host_srv_serror_rate
19	Num_access_shells	40	Dst_host_rerror_rate
20	Num_outbound_cmds	41	Dst_host_srv_rerror_rate
21	Is_not_login		

IV. EXPERIMENTAL RESULTS AND EVALUATIONS

In this section, we discuss the experimental results of our proposed DDoS mitigation technique. In the following subsections, we discuss the experimental framework and the evaluation metrics for the proposed scheme. Then we discuss the yielded results in detail.

A. Experimental Framework

As soon as the attack starts as shown in Figure 5, the SDN controller on ISP 1 will forward a statistical information about the network sessions to the attack detecting server, the attack detecting server will be able to identify these sessions as DDoS or a normal traffic.

If the attack detecting server identifies it as a potential DDoS attack, the SDN controller notifies the SDN controller on the attacker's ISP about the attack with the attack source's IP address, as shown in Figure 6. On the other hand, if the sessions are identified as normal traffic, then an IP based flow will be installed. After that, the SDN controller on the attacker's ISP needs to verify that the traffic is really a DDoS and not a benign traffic. If the traffic was a DDoS, it will block the source of that traffic.

In our proposed model, we use 25 features as input where the result is based on these features. The result can be normal or abnormal traffic. Then the controller decides whether to forward or to drop a packet.

B. Evaluation Metrics

We evaluate the proposed DDoS mitigation mechanism from precision, recall, and F1 score perspective. Precision is the ratio of the sessions that were truly identified as attacks and the overall predicted sessions (whether true or false). On the other hand, recall is the ratio of the truly identified sessions and the number of sessions that may have caused the attack and must have been identified. Both precision and recall collectively define the accuracy of the algorithm through F1 score. The F1 score is generally the harmonic average of the precision and recall. When F1 score reaches 1, it means accurate measurement of precision and recall, and it reaches zero when the precision and recall are incorrect. The accuracy of the proposed model can be seen in the Table II.

TABLE II
TESTING ON THE NSL-KDD DATASET

	Precision	Recall	F1-score
anomaly	0.02	1.00	0.04
normal	0.99	0.96	0.98
avg/total	0.98	0.98	0.98

After checking the accuracy on the afore-mentioned dataset, we use a real DDoS attack on the model and analyze the accuracy. At first, we attack the system using SYN flooding. The accuracy can be seen in Table Table III.

TABLE III
RESULTS AFTER ATTACKING

	Precision	Recall	F1-score
anomaly	1	0.5	0.66
normal	0	0	0
avg/total	0.5	0.5	0.66

It is worth noting that there are some useless and redundant features in the features set and we test if the accuracy could be improved by removing those features. After trial and error, we remove the following features from the model and train the model again. With the new features set, we obtained the results as shown in Table IV.

- Is_guest_login
- Src_byte
- Rerror_rate
- Srv_rerror_rate
- Num_root
- Dst_host_srv_serror_rate
- Num_access_shells

TABLE IV
RESULTS AFTER REMOVING SOME FEATURES

	Precision	Recall	F1-score
anomaly	0.92	0.66	0.76
normal	0.67	0.93	0.78
avg/total	0.81	0.77	0.77

Next we want to test the number of sessions that the proposed solution can handle. Therefore, we setup a testing environment to analyze the number of sessions as well as the time required to gather the data to extract features. These features are to be sent to ML server and we analyze the time required for ML server to detect if that traffic is a DDOS or not, and finally the communication time between the controllers and the ML server. The results are shown in Table V.

TABLE V
NUMBER OF SESSIONS WITH THE MATCHING TIME

Number of sessions	Time to gather data	Time to detect	Total Time
1000	1.5 second	0.6 second	2.1 second
10,000	3 seconds	0.9 seconds	3.9 seconds
100,000	5 seconds	1.3 second	6.3 seconds

Finally we tested the CPU usage at the attacker, target and the ML server to check if the proposed solution is lightweight or does it require a lot of CPU power. During the tests, we downloaded a number of files, did Web surfing, watched online videos and used iperf3 tool to reach the max throughput of the proposed solution. The obtained results are shown in Table VI.

TABLE VI
CPU USAGE

Traffic Size	CPU usage of Target controller	CPU usage of Attacker controller	CPU usage of the ML server
100Mbps	5%	1%	10%
1000Mbps	25%	5%	45%
3000Mbps	75%	20%	90%

V. CONCLUSION AND FUTURE WORK

Mitigating DDoS attacks in SDN is of prime importance for harnessing its capabilities. In this paper, we combined the programmability power of SDN with the intelligence of ML to mitigate DDoS attacks in SDN. We used NSL-KDD dataset

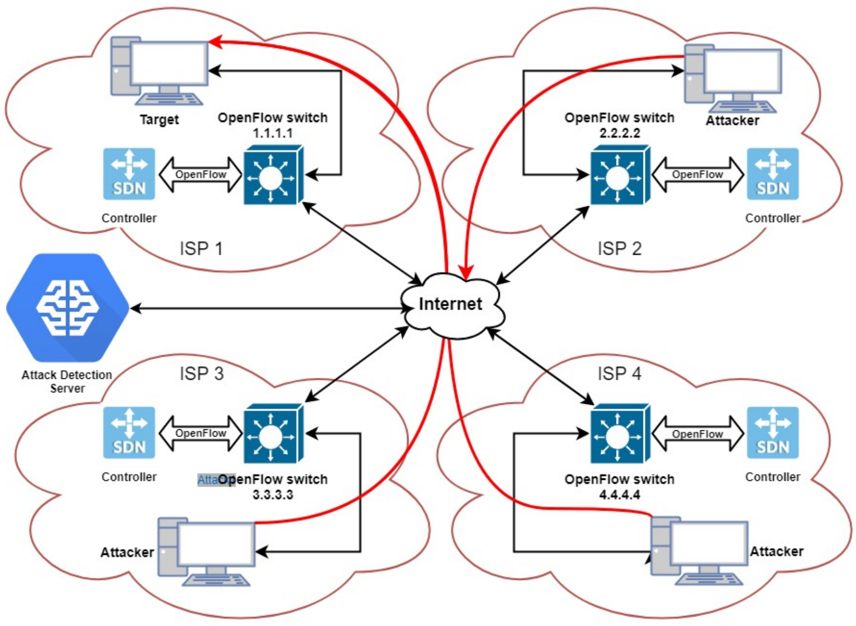

Fig. 5. Attack scenario

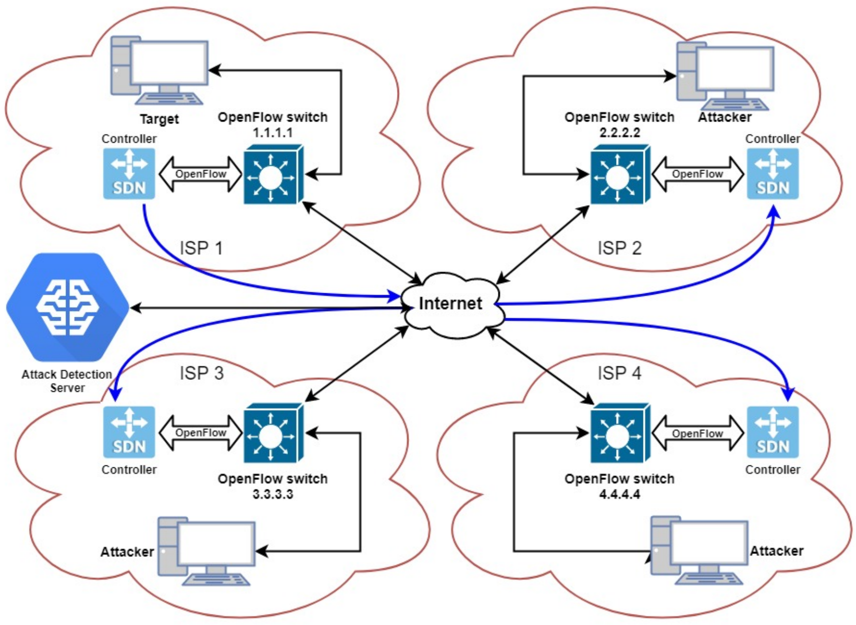

Fig. 6. The SDN controller notify other controllers

to train our designed model for DDoS mitigation and then tested the trained model on real DDoS attack. As a result, we achieved encouraging results for DDoS mitigation in SDN.

Additionally, we implemented a communication protocol between the SDN controllers in order create a faster and more reliable DDoS mitigation method. The preliminary results are promising and subject to further investigation. In the future, work we will try to cover more network attacks type and try to implement network anomaly detection. Furthermore, we also plan to detect data breaches in addition to DDoS attacks in SDN.

References

[1] Sandhya, Y. Sinha, and K. Haribabu, "A survey: Hybrid sdn," *Journal of Network and Computer Applications*, vol. 100, pp. 35 – 55, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S108480451730317X

[2] M. Conti, A. Gangwal, and M. S. Gaur, "A comprehensive and effective mechanism for ddos detection in sdn," in *2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct 2017, pp. 1–8.

[3] S. Das, Y. Liu, W. Zhang, and M. Chandramohan, "Semantics-based online malware detection: Towards efficient real-time protection against malware," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 2, pp. 289–302, Feb 2016.

[4] J. Moon and Y. Lim, "Access control of mtc devices using reinforcement learning approach," in *2017 International Conference on Information Networking (ICOIN)*, Jan 2017, pp. 641–643.

[5] A. Alshamrani, A. Chowdhary, S. Pisharody, D. Lu, and D. Huang, "A defense system for defeating ddos attacks in sdn based networks," in *Proceedings of the 15th ACM International Symposium on Mobility Management and Wireless Access*, ser. MobiWac '17. New York, NY, USA: ACM, 2017, pp. 83–92. [Online]. Available: http://doi.acm.org/10.1145/3132062.3132074

[6] S. Khandelwal, "World's largest 1 tbps ddos attack launched from 152,000 hacked smart devices," https://thehackernews.com/2016/09/ddos-attack-iot.html, 2016.

[7] S. Hilton, "Dyn analysis summary of friday october 21 attack," https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/, 2016.

[8] T. KNECHT, "5 biggest ddos attacks of the past decade," https://www.abusix.com/blog/5-biggest-ddos-attacks-of-the-past-decade, 2017.

[9] S. Hameed and H. A. Khan, "Leveraging sdn for collaborative ddos mitigation," in *2017 International Conference on Networked Systems (NetSys)*, March 2017, pp. 1–6.

[10] X. Yang, B. Han, Z. Sun, and J. Huang, "Sdn-based ddos attack detection with cross-plane collaboration and lightweight flow monitoring," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Dec 2017, pp. 1–6.

[11] J. Zheng, Q. Li, G. Gu, J. Cao, D. K. Y. Yau, and J. Wu, "Realtime ddos defense using cots sdn switches via adaptive correlation analysis," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 7, pp. 1838–1853, July 2018.

[12] C. Gkountis, M. Taha, J. Lloret, and G. Kambourakis, "Lightweight algorithm for protecting sdn controller against ddos attacks," in *2017 10th IFIP Wireless and Mobile Networking Conference (WMNC)*, Sept 2017, pp. 1–6.

[13] N. Meti, D. G. Narayan, and V. P. Baligar, "Detection of distributed denial of service attacks using machine learning algorithms in software defined networks," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Sept 2017, pp. 1366–1371.

[14] L. Barki, A. Shidling, N. Meti, D. G. Narayan, and M. M. Mulla, "Detection of distributed denial of service attacks in software defined networks," in *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Sept 2016, pp. 2576–2581.

[15] J. Ashraf and S. Latif, "Handling intrusion and ddos attacks in software defined networks using machine learning techniques," in *2014 National Software Engineering Conference*, Nov 2014, pp. 55–60.

[16] "Ryu official site," https://osrg.github.io/ryu.

[17] "Pox manual," https://noxrepo.github.io/pox-doc/html.

[18] "Onos official site," https://onosproject.org.

[19] "Opendaylight official site," https://www.opendaylight.org.

[20] "Nsl-kdd dataset," *http://www.unb.ca/cic/datasets/nsl.html*, 2015.

[21] D. S. S. L.Dhanabal, "A study on nsl-kdd dataset for intrusion detection system based on classification algorithms," *International Journal of Advanced Research in Computer and Communication Engineering*, 2015.