Mitigating DNS Query-Based DDoS Attacks with Machine Learning on Software-Defined Networking

Muhammad Ejaz Ahmed, Hyoungshick Kim, and Moosung Park

***Abstract*—Securing Internet of Things is a challenge because of its multiple points of vulnerability. In particular, Distributed Denial of Service (DDoS) attacks on IoT devices pose a major security challenge to be addressed. In this paper, we propose a DNS query-based DDoS attack mitigation system using Software-Defined Networking (SDN) to block the network traffic for DDoS attacks. With some features provided by SDN, we can analyze traffic patterns and filter suspicious network flows out. To show the feasibility of the proposed system, we particularly implemented a prototype with Dirichlet process mixture model to distinguish benign traffic from malicious traffic and conducted experiments with the dataset collected from real network traces. We demonstrate the effectiveness of the proposed method by both simulations and experiment data obtained from the real network traffic traces.**

1. Introduction

The defense and intelligence communities face a growing demand to accelerate the deployment of new applications and networks against sophisticated cyber attacks. The unmatched potential of such cyber attacks, including espionage, military and strategic data stealing, Distributed Denial of Service (DDoS) attack, or even control on command and control systems, is a major concern for defense communities. Making it worse, the dramatic increase in the Internet of Things (IoT) systems (as Gartner reports in [1] that the number of internet-enabled IoT devices will increase to 26 billion by 2020) with the associated vulnerabilities could be exploited to launch massive DDoS attacks against any military organization. For example, in 2016, a massive DDoS attack launched by the *Mirai* botnet was carried out by compromised Internet-enabled IoT devices to target the Dyn server which knocked offline major websites including Twitter, Spotify, Amazon, Reddit, Netflix and The New York Times [2]. In particular, the proliferation of vulnerable IoT devices is yet another big challenge that military organizations have to consider to defend against DDoS attacks.

Due to the multifaceted nature of the problem, such as lack of stringent security measures in IoT systems, DNS protocol exploitation, and limitations of packet forwarding in traditional networks, it is a big challenge for a military organization to mitigate such attacks effectively. The detailed description of the problems mentioned above is as follows:

- **Malware:** Malware can be used for searching the Internet for vulnerable IoT devices to compromise them by exploiting the identified vulnerabilities. Moreover, the lack of proper security measures (e.g., misuse of factory default passwords) allows attackers to access IoT devices in an unauthorized manner. For example, the list of 68 usernames and passwords was used for the *Mirai* botnet [2].
- **Exploitable DNS protocol:** The DNS protocol can be used to launch a DDoS attack by sending relatively small queries (e.g., about 60 bytes) with a spoofed source address (i.e., the victim's address) to DNS resolvers. As a result, DNS resolvers reply with significantly larger responses (e.g., about 4,000 bytes) to exhaust the victim's resources [3]. Unlike conventional DDoS attacks, however, blacklisting the IP addresses of the open DNS resolvers is not acceptable because it might affect legitimate DNS resolutions.
- **Limitations of the Internet:** Traditional network infrastructure relies on the routing table lookup routine to forward packets from input to output ports on the packet path. However, due to the network resources' stringent memory and computational requirements, it is not possible for an in-line (in the line of direct communication) real-time anomaly detection systems (ADSs) to examine every packet in detail. It is therefore a job of the targeted organization's firewall/ADS to examine traffic flows in detail to find suspicious traffic. Moreover, in some environments, the ingress filtering is disabled. Therefore attackers can conceal their network identities using IP address spoofing and reflection.

Wang et al. [4], [5] have conducted analysis on a large scale DDoS dataset in order to understand the dynamics behind an increasingly sophisticated DDoS attack strategies. Tuan et al. [6] proposed a deep learning approach for network intrusion detection in Software-Defined Networking (SDN). They constructed a simple deep neural network with an input layer, three hidden layers and an output layer. Braga et al. [7] proposed a lightweight method for detecting DDoS attacks based on the traffic flow features, in which the extraction of such information is made with a very low overhead compared with existing solutions [6], [8]. Akbar et al. [9] demonstrated that an SDN-based network router can effectively be used for detecting security threats in SOHO

This work was supported by Defense Acquisition Program Administration and Agency for Defense Development under the contract (UD060048AD).
M. E. Ahmed and H. Kim are with the College of Software, Sungkyunkwan University (SKKU), Suwon, Korea (e-mail: ejaz629@skku.edu, hyoung@skku.edu) M. Park is with Agency for Defense Development, Korea (e-mail: parkms@add.re.kr).

(Small Office/Home Office) networks.

In this paper, we extend their proposals into a more advanced SDN architecture to examine network traffic flows on packet path and drop malicious flows/packets generated for DDoS attacks. Since the SDN controller can not only maintain a global view of the entire network but also dynamically manage switches in an SDN environment, it seems effective to develop a DDoS mitigation solution for detecting separated suspicious traffic flows in a comprehensive manner. To show the feasibility of the proposed system, we implemented a prototype based on traffic features using Dirichlet process mixture model (DPMM) to cluster various traffic applications flows including the traffic flows used for DDoS attacks in an unsupervised manner.

The proposed DPMM is a nonparametric Bayesian approach for clustering traffic applications (patterns) where the nonparametric means that the number of traffic applications are unknown and may grow over time. In the proposed scheme, SDN controller examines the traffic at the underlying network devices using the OpenFlow protocol. Our approach differs from other clustering algorithms in the following ways:

1) Network traffic can be monitored and analyzed with minimum overhead in network resources such as router and switch because the SDN controller can periodically request for traffic statistics from the underlying network resources rather than continuously observing using a dedicated control channel.
2) The proposed system can flexibly be adapted to various network situations. More traffic features can be added to increase the accuracy of clustering algorithm. Also, any prior knowledge about the number of different traffic applications is not required.

We model the feature space of a single traffic cluster as a multivariate Gaussian distribution with unknown parameters. For multiple traffic clusters, we model it as infinite Gaussian mixture model known as DPMM. The Dirichlet distribution is used to define the prior, the collapsed Gibbs sampling algorithm [11] is used in conjunction with Dirichlet prior to estimate the number of clusters (applications) based on the Chinese restaurant process.

The rest of the paper is organized as follows. Section 2 describes a brief introduction about DNS query-based DDoS attacks. Section 3 introduces the SDN architecture. Section 4 explains a SDN-based DPMM clustering to mitigate DDoS attacks. Section 5 presents the experiment results. In Section 6, we conclude the study.

2. DNS query-based DDoS attacks

In this section, we discuss two representative types of DNS query-based DDoS attacks.

2.1. DNS amplification

To launch a DNS amplification attack, attackers send high volumes of forged DNS queries with a spoofed IP address (i.e., the victim's address) to a large number of open DNS resolvers to prompt them to send massive volumes of DNS query responses to the victim with that address. With hundreds of such fake queries being sent out from the compromised IoT devices and with several DNS resolvers replying back simultaneously, the victim organization's server can easily be overwhelmed by the huge number of DNS responses. To amplify a DNS attack, each DNS request can be sent using the *EDNS0* DNS protocol extension, which allows for large DNS messages, or using the cryptographic feature of the DNS security extension such as *DNSSEC* to increase the message size. Spoofed queries of the type "ANY," which returns all known information about a DNS zone in a single request, can also be used. Using these methods, a DNS request message of some 60 bytes can be configured to trigger a response message of over 4000 bytes to the targeted organization's server, resulting in a 70:1 amplification factor. This markedly increases the volume of traffic the targeted server receives, and accelerates the rate at which the servers resources will be depleted.

2.2. DNS flooding

Unlike DNS amplification, DNS flooding attacks are symmetrical DDoS attacks. These attacks attempt to exhaust server-side resources with a flood of UDP requests, generated by scripts running on several compromised botnet devices. To attack a DNS server with a DNS flood, the attacker runs a script, generally from multiple servers. These scripts send malformed packets from spoofed IP addresses. Since Layer 7 attacks like DNS flood require no response to be effective, the attacker can send packets that are neither accurate nor even correctly formatted. Another common type of DNS flood attack is *DNS NXDOMAIN* flood attack, in which the attacker floods the DNS server with requests for records that are nonexistent or invalid. The DNS server expends all its resources looking for these records, its cache fills with bad requests, and it eventually has no resources to serve legitimate requests.

3. SDN architecture

The SDN architecture decouples the network control from the data (forwarding functions) which enables the network control to become directly programmable. This separation between the control and data planes provide flexibility and control to organizations to better manage their network devices and dynamically add/updates policies. Fig. 1 illustrates network infrastructure for traditional and SDN networks. We can see that the SDN controller manages its network devices, i.e., OpenFlow-switches (OF-switches), using the OpenFlow protocol, represented by the red dashed arrows. Each OF-switch contains flow tables which holds statistics about the traffic passing through it. A flow table entry is illustrated in Fig. 2. The OpenFlow protocol [10] allow OF-switches to be managed by an external controller. On packet arrival at OF-switch, if a flow entry for the packet exists in a flow table, it is forwarded in a normal way

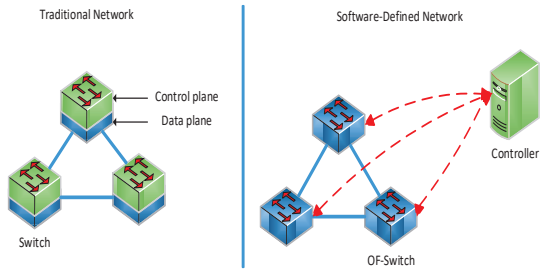


Figure 1. Traditional network architecture vs. SDN network architecture.

otherwise the incoming packet is sent to the controller for further analysis. The controller may perform the following actions: install flow entry for traffic flow in the OF-switch, drop traffic flows from malicious sources, and mirror any port to the external server for thorough packet analysis.

Surely, SDN fits well with military networks because all network resources in a military network can be monitored and/or managed centrally by a single control plane. We summarize the features of SDN in the following sections.

3.1. Decoupled control and data planes

SDN decouples the data and control planes, thereby providing grounds to establish large scale attack and defense experiments. Progressive deployment of innovative ideas can be realized via seamless transition from an experimental phase to an operational phase. Moreover it enables innovation and evolution by providing a programmable network platform to implement, experiment, and deploy new ideas and new applications. By using this feature, DDoS attacks (e.g., DNS amplification or DNS flooding) detection and mitigation could be realized with much convenience in a military organization's network.

The controller has network-wide knowledge of the system and a global view to build consistent security policies and to monitor/analyze traffic patterns for potential security threats. SDN controllers monitor traffic flows from various sources with-in their respective network domains and identify potential anomalies in the traffic behavior.

3.2. Dynamic updating of forwarding rules

The OpenFlow protocol provides a common interface to control how packets are forwarded by accessing OF-switch's internal flow tables and the statistics about those flows. A flow is a 12-tuple entry in a flow table in OF-switches with the fields that are matched against the incoming packets. Each flow entry has an action and a statistics field, as illustrated in Fig. 2. Each flow entry contains a header against which the incoming packets are matched and the corresponding action is applied on the packet. On the other hand, if no matching entry is found in the flow table of an OF-switch, the packet is forwarded to the controller. After analyzing those packets, the controller may decide to install flows in the OF-switch or just tell the OF-switch to drop the

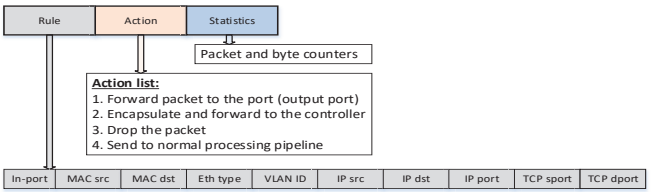


Figure 2. Flow table entry.

packet. Additionally, controllers can perform the following after analyzing the packet: 1) Add flow entry in the switch, 2) Drop packets from the flow it is malicious, 3) Mirror any port to external server for flow analysis.

3.3. Software-based traffic analysis

Software-based traffic analysis greatly enables innovation, as it is possible to improve the capabilities of a switch using any software-based technique. Traffic analysis can be performed in real time using machine learning algorithms (Support Vector Machines (SVMs), Gaussian Mixture Models (GMM), Artificial Neural Networks (ANNs)), databases and any other software tool. Traffic of interest can be explicitly directed to intrusion prevention systems (IPSs) for Deep Packet Inspection (DPI).

4. SDN-based DPMM clustering

We briefly present our system for monitoring and detection of attacks in SDN environments. The proposed system consists of the following components: traffic statistics manager, learner component and network resources manager.

The traffic statistics manager collects traffic features from the underlying network devices. The collected traffic features such as packet length/duration, connection time, uplink/downlink traffic play an important role in designing traffic classification algorithms. The SDN controller can monitor traffic flows at all OF-switches in its administrative domain and gather network traffic statistics when needed. An *ofp_flow_stats_request* message is sent by the controller to every OF-switch after a fixed time period to request the network traffic statistics. OF-switches reply back to the controller with an *ofp_flow_stats_reply* message with the requested statistics. Thus, the controller can take advantages of the complete network view provided by the SDN to analyze this feedback from the network.

The learner component is responsible for the detection of attack traffic flows on the network devices. In general, machine learning algorithms used for classification or clustering are implemented for the learner component. In this paper, we implemented a prototype using DPMM clustering algorithm that can be deployed in the learner component in SDN controller. All network traffic statistics are sent to the DPMM module in the learner component to analyze and detect suspicious network traffic. Once a network anomaly is detected and identified, the OpenFlow protocol can effectively mitigate it by configuring a rule to block such traffic at switches.

The network resources manger keeps track of network devices' status in terms of CPU and memory utilization.

4.1. Traffic features selection

In our experiments, we consider the following three features of a connection: total number of packets transmitted (ToP), ratio of source and destination bytes (RoSD), and connection duration time (CT). Then, the ToP is represented as $T_{\text{P}} = \{ToP_1, ToP_2, \cdots, ToP_N\}$, where ToP_n is the number of packets transmitted in the n^{th} traffic connection. Similarly, RoSD and CT are represented as $R_{\text{B}} = \{RoSD_1, RoSD_2, \cdots, RoSD_N\}$ and $T_{\text{C}} = \{CT_1, CT_2, \cdots, CT_N\}$, respectively. Let $\mathbf{x}_n$ be the feature vector of the nth connection, given by

$$\begin{aligned} \mathbf{x}_n &= \big[ToP_n, RoSD_n, CT_n\big], \\ \mathbf{X} &= \big[\mathbf{x}_1, \cdots, \mathbf{x}_N\big]^T, \end{aligned} \tag{1}$$

where N be the number of observations (connections). The matrix of observations $\mathbf{X}$ are observed by the SDN controller using the request/reply messages, discussed above. Given the observations $\mathbf{X}$ from N traffic flows (connections), the proposed DPMM module finds how many legitimate (e.g., HTTP, FTP, etc.) and attack traffic connections are active over N traffic connections.

4.2. Dirichlet Process Mixture Model

Finite Gaussian mixture model (FGMM) is a hidden-variable probabilistic model formed by the sum of weighted Gaussian random variables. The FGMM is frequently used since it can approximate an arbitrary multi-modal probability density function (*p.d.f.*) and cluster data if the hidden variables are interpreted as classes labels. The FGMM can be expressed as

$$P(\mathbf{x}_n|\Theta, \vec{\pi}) = \sum_{k=1}^{K} \pi_k p_k(\mathbf{x}_n|\theta_k) \tag{2}$$

where $p_k(\mathbf{x}_n|\theta_k)$ is the *p.d.f.* of the kth class (traffic flow) and is multivariate Gaussian with parameter θ_k, π_k is the mixing proportion or class prior, $\Theta = \{\theta_k\}_{k=1}^{K}$ is the collection of all class parameters with $\theta_k = \{\vec{\mu}_k$ and $\Sigma_k\}$ where $\vec{\mu}_k$ and Σ_k are the mean and covariance for class k, and $\vec{\pi} = \{\pi_k\}_{k=1}^{K}$ are the Gaussian-mixture weights. The graphical model of FGMM is represented in Fig. 3. We assume that $\pi_k \geq 0$ for $k \in \{1, \cdots, K\}$ and $\sum_{k=1}^{K} \pi_k = 1$. Let $\pi_k = p(z_n = k)$ represent the prior probability that the data point was generated from the kth traffic flow, where z_n indicates which traffic application (HTTP, FTP, or attack traffic) the data point $\mathbf{x}_n$ belongs to. The graphical model of DPMM is represented in Fig. 3. The generative model of DPMM is defined as

$$\begin{aligned} \vec{\pi}|\alpha &\sim \text{Stick}(\alpha), \quad z_n|\vec{\pi} \sim \text{Multinomial}(\cdot|\vec{\pi}), \\ \Sigma_k &\sim \text{Inverse Wishart}_{\upsilon_0}(\Lambda_0), \quad \vec{\mu}_k \sim \text{G}(\vec{\mu}_0, \Sigma_k/k_0), \\ \mathbf{x}_i|z_i = k, \Sigma_k, &\quad \vec{\mu}_k \sim G(\cdot|\vec{\mu}_k, \Sigma_k), \end{aligned} \tag{3}$$

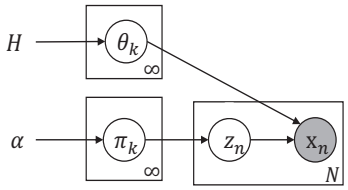


Figure 3. Graphical model of a Bayesian DPMM. Circles indicate random variables and boxes (plates) indicate the repetition of the random variables in boxes. The links among random variables represent the dependency among them.

where $\vec{\pi}|\alpha \sim \text{Stick}(\alpha)$ is,

$$\pi'_k|\alpha \sim \text{Beta}(1, \alpha), \;\; \pi_k = \pi'_k \prod_{l=1}^{K-1} (1 - \pi'_k), \;\; K \to \infty.$$

Since the DPMM is the generative model, we need conjugate priors for the observations which is a multivariate normal distribution. The conjugate prior of multivariate normal is the inverse-Wishart distribution. In DPMM, the number of classes goes to infinity; therefore, it is difficult to model $\vec{\pi}$ with the Dirichlet distribution. Hence, the *Stick breaking construction* [11], is used to model the class weights. In stick breaking process, a stick of length 1 is broken into two parts, which is modeled as a *Beta* distribution. The length of broken stick becomes the weight, π_1, this process is repeated.

We are interested in finding the posterior distribution over $Z = \{z_n\}_{n=1}^{N}$. By exploiting the dependencies among random variables, and applying Bayes' rule and conditioning on the observed data and hyperparameters, the posterior is given as

$$\begin{aligned} P(Z, \Theta, \vec{\pi}, \alpha|\mathbf{X}; \mathcal{H}) \propto & \prod_{n=1}^{N} P(\mathbf{x}_n|z_n, \theta_{z_n}) \prod_{k=1}^{K} P(\theta_k; \mathcal{H}) \\ & \prod_{n=1}^{N} P(z_n|\vec{\pi}) \cdot P(\vec{\pi}|\alpha) P(\alpha) \end{aligned} \tag{4}$$

where $\mathbf{x}_n$ are the observations (shaded circle in Fig. 3). By integrating $\vec{\pi}$ out, and as $K \to \infty$, the posterior distribution in (4) becomes

$$P(z_n = k|\mathbf{X}, Z_{-i}, \alpha; \mathcal{H}) \propto P(z_n = k|Z_{-i}, \alpha)\, P(\mathbf{x}_n|\mathbf{X}_{k,-i}; \mathcal{H}). \tag{5}$$

The problem of model selection in the FGMM can be avoided by assuming that there are an infinite number of classes (leads to DPMM) but only a finite number of feature points are observed. The Chinese Restaurant Process (CRP) is the process that generates samples from such a model. In such case, the MCMC methods such as Gibbs sampler can be used for a discrete representation of the posterior by sampling from the unnormalized *p.d.f.* in (4). For details about CRP process and MCMC based Gibbs sampler, please refer to [11].

The clustering result will assign label z_n to each traffic flow (connection) with a positive integer value. Unique traf-

TABLE 1. Summary of the measured traffic data.

Dataset		
Flows		113,193
TCP connections		82,095
Protocol composition	HTTP	87.4%
	FTP	6.48%
Data	Source Bytes	311,170,654
	Destination Bytes	4,148,023,048
Packets	Source packets	2,340,871
	Destination packets	3,599,451
Traffic	Normal	131,111
	Attack	2,082

fic applications are assigned different labels. Attack traffic connections are also assigned a label which is same for all attack traffic. After the clustering process, the number of clusters obtained may vary depending upon the number of unique traffic applications and attack traffic, i.e., feature points, $\mathbf{x}_n$, from legitimate traffic flows (applications) concentrate at a single location in three dimensional feature space and constitute a single cluster; however, attack traffic flows form a separate cluster in three dimensional feature space, as shown in Fig. 4, to distinguish DDoS attack from legitimate traffic applications.

5. Experiment results

To study the effectiveness of the proposed DPMM clustering approach for attack detection, we evaluate the performance in the following aspects:

- **Attack traffic classification accuracy:** The measure of classifying the attack traffic connections correctly. It is the ratio of number of traffic connections correctly classified to the total number of attack traffic connections.
- **Overall classification accuracy:** The measure of classifying each feature vector to its correct cluster (application). It is defined as the number of feature vectors assigned to their original clusters (applications) over the total number of feature vectors.
- **Misclassification rate:** It is the number of feature vectors assigned to the wrong clusters (applications) over the total number of feature vectors.

5.1. Implementation considerations

In this set of experiments, we use real traffic traces with attack scenarios in [13]. The description of the dataset is given in Table 1. We run the program on dual core i7-4790 CPU with speed 3.60 GHz each core, and the installed memory of 8 GB. In the set of experiments, the number of TCP connections vary from 200 to 500 including legitimate and attack traffic. The dataset we used in our experiments contained more than 80,000 TCP connections. The dataset consists of 3 classes (HTTP: about 87%, FTP: 6%, attack: 7%). For clustering, we randomly picked 100 TCP connections for each application and extract features of them to input to the proposed DPMM module in the SDN controller.

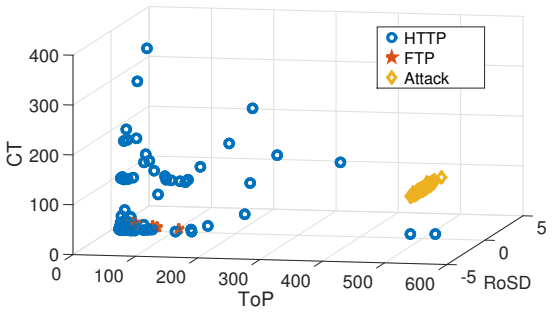


Figure 4. Three dimensional visualization of HTTP, FTP, and attack traffic.

We also generated synthetic traffic flow features for malicious traffic flows. For that, we generated each traffic flow feature following *Normal* random variable with mean and variance set in reference to the real dataset. We generated the features (ToP, RoSD, CT) for attack traffic with the following distribution parameters, $\mathcal{N}(500, 2)$, $\mathcal{N}(0.15, 1)$, $\mathcal{N}(100, 1)$, respectively. In DPMM, to obtain the posterior distribution over cluster labels Z, we need to set the hyperparameters: $\alpha, \vec{\mu}_0, \kappa_0, \Lambda_0^{-1}$ and ν_0. The hyperparameters are set to $\mathcal{H} = \{\Lambda_0^{-1}, \nu_0, \vec{\mu}_0, \kappa_0\} = \{\text{Identity}(3), 4, \text{Zeros}(3, 1), 0.1\}$. α encodes the number of clusters.

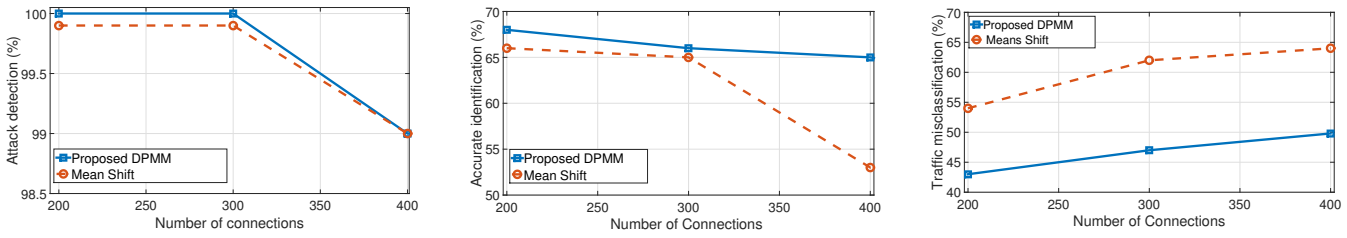
5.2. Experiments

We compare the proposed DPMM clustering approach with the mean-shift (MS) approach [14]. The MS is a nonparametric classification method which treats the physical location density of feature points as the probability density. The dense regions represent the means of underlying probability densities. The idea is to locate the dense regions in the data set to find the number of clusters.

Fig. 4 show traffic flow features in a 3-dimensional feature space. We can observe that the HTTP traffic shows a lot of variation in its feature space. However, FTP traffic is concentrated in a specific location in the graph. The reason for the variations in the HTTP traffic is due the request-response pattern which is mostly unique for each user. Such large variations in features can affect the classification accuracy and may result in false positives. However, the FTP and attack traffic are concentrated at different location and are fairly apart from each other in a 3-dimensional feature space.

The performance results are shown in Fig. 5. For improved visualisation, we use the same range on the y-axis for overall classification accuracy and misclassfication rate only since the level of attack traffic classification accuracy is totally different from those metrics.

Our first performance metric is attack traffic classification accuracy. Fig. 5a demonstrates the accuracy of the proposed DPMM clustering and MS approaches. It can be observed that the accuracy of detecting a attack traffic connection reduces as the number of traffic flows (legitimate



(a) Attack traffic classification accuracy (b) Overall classification accuracy (c) Misclassification rate

Figure 5. Performance comparison of the proposed DPMM with the Means Shift clustering algorithms.

and attack) increases. Here, our main focus is to detect malicious traffic flows correctly. We observe that the both the approaches performs well in detecting malicious flows. However, in identifying traffic application of benign traffic flows, the difference between these two approaches is quite noticeable.

For overall traffic classification accuracy, Fig. 5b show the results for correctly identifying legitimate and attack traffic connections. We observe that the proposed DPMM outperforms the MS algorithm in correctly identifying HTTP, FTP, and attack traffic. Moreover, MS algorithm struggles in identifying FTP traffic pattern and misclassify it as HTTP traffic. In other words, MS algorithm performs poorly in detecting FTP traffic and all the connections from FTP are labeled as HTTP, therefore, the misclassification rate for FTP is higher than the proposed DPMM. Our final performance metric is the misclassification rate, we can observe from Fig. 5c that the proposed DPMM approach performs significantly better than the MS algorithm in misclassification rate. It is evident from Fig. 5c that the proposed DPMM misclassification rate is less than 50% for all traffic connections, where as for MS, the misclassification rate is greater than 54%. Thus the proposed DPMM approach not only perform well in identifying DDoS attack but also in detecting various traffic flows such as HTTP and FTP.

6. Conclusion

In this paper, we demonstrated that SDN can be used to mitigate DDoS attacks with a global view of the entire network. Naturally, DDoS attacks cannot easily be prevented by a few network entities alone because each network entity such as router has only local knowledge about some particular paths and neighbor nodes. The proposed system seems an alternative architecture to address such concerns. In particular, we note that the proposed model is suitable for some environments (e.g., military networks) with a centralized controller because all network resources can be monitored and/or managed in a centralized infrastructure.

To show the feasibility of the proposed framework, we implemented a prototype using the Dirichlet process mixture model for clustering traffic flows. The proposed algorithm outperformed another nonparametric mean shift (MS) clustering method not only in accurately identifying DDoS attack traffic flows but also in identifying the traffic flows for two most popular network applications such as HTTP and FTP.

As part of our future work, we plan to implement the proposed mitigation technique and deploy it in a real network environment and further investigate the performance of the proposed system on a large scale.

References

[1] C. Pettey, "The Internet of Things and the Enterprise," Gartner, Aug. 2015.

[2] B. Krebs, "Who Makes the IoT Things Under Attack?," KrebsonSecurity, Oct. 2016.

[3] [Online] DNS Amplification Attack, Incapcula, Feb 24, 2017.

[4] A. Wang, A. Mohaisen, W. Chang, and S. Chen, "Capturing DDoS attack dynamics behind the scenes," *In International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pp. 205-215, Jul. 2015.

[5] A. Wang, A. Mohaisen, W. Chang, and S. Chen, "Delving into internet DDoS attacks by botnets: characterization and analysis," *In 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 379-390, Jun. 2015.

[6] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep Learning Approach for Network Intrusion Detection in Software Defined Networking," *IEEE Proc. WINCOM,* pp. 258-263, Fez, Morocco, Oct. 2016.

[7] R. Braga, M. Edjard, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," *IEEE Proc. LCN,* pp. 408-415, Denver, Colorado, U.S.A., Oct. 2010.

[8] W. Wang and S. Gombault, "Efficient Detection of DDoS Attacks with Important Attributes," *Proc. CRISIS,* Denver, Tozeur, Tunisia, Oct. 2008.

[9] S. A. Mehdi, J. Khalid, and S. A. Khayam, "Revisiting traffic anomaly detection using software defined networking," *Springer International Workshop on Recent Advances in Intrusion Detection,* pp. 161-180, Berlin Heidelberg, 2011.

[10] OpenFlow, "OpenFlow Switch Specification, Version 1.5.1," *Open Networking Foundation,* Mar. 2015.

[11] D. Blei and M. Jordan, "Variational inference for Dirichlet process mixtures," *Bayesian Analysis*, vol. 1, no. 1, pp. 121-144, Aug. 2006.

[12] N. T. Nguyen, R. Zheng, and Z. Han, "On Identifying Primary User Emulation Attacks in Cognitive Radio Systems Using Nonparametric Bayesian Classification," *IEEE Transactions on Signal Processing*, vol. 60, no.3, p.p. 1432-1445, March 2012.

[13] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers and Security,* vol. 31, no. 3, pp. 357-374, 2012.

[14] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE transactions on pattern analysis and machine intelligence*, vol. 17, no. 8, pp. 790-799, 1995.