Statistical Application Fingerprinting for DDoS Attack Mitigation

Muhammad Ejaz Ahmed, Saeed Ullah, and Hyoungshick Kim

***Abstract*—Because of the dynamic nature of network traffic patterns, such as new traffic application arrivals or flash events, it is becoming increasingly difficult for conventional anomaly detection systems to separate various applications based on their traffic patterns. In this study, by leveraging transport layer packet-level and flow-level features, new structures called *application fingerprints* are generated, which express such features in a compact and efficient manner. Based on the generated fingerprints, we propose a novel traffic classification framework. The proposed system generates profiles of normal applications using a multi-modal probability distribution. The proposed classification framework is then extended to detect distributed denial of service (DDoS) attacks from the collected statistical information at flow level. To demonstrate the feasibility of the proposed system, we evaluate its performance using five real-world traffic datasets. The experiment results show that the proposed method is capable of achieving an accuracy of over 97%, whereas the misclassification rate is only 2.5%.**

***Index Terms*—Traffic classification; Anomaly detection; Fingerprinting traffic applications; Nonparametric clustering; DDoS attack detection.**

I. Introduction

THE unmatched potential of cyber-attacks, including espionage, military and strategic data stealing, distributed denial of service (DDoS), or even control attacks on command and control systems, is a major concern in the modern era. Making this worse is the dramatic increase in the Internet of Things (IoT) systems with their associated vulnerabilities, which could be exploited to launch massive DDoS attacks against any organization. Gartner reports that the number of internet-enabled IoT devices will increase to 26 billion by 2020 [1]. For example, in 2016, a massive DDoS attack in the range of 1.2 Tbps [2] launched by the *Mirai* botnet was carried out using compromised Internet-enabled IoT devices to target the Dyn server, which knocked major websites offline including Twitter, Spotify, Amazon, Reddit, Netflix, and The New York Times [3]. In particular, the proliferation of vulnerable IoT devices presents yet another significant challenge that organizations must consider in order to defend against DDoS attacks. As a result, organizations must have to protect themselves against such attacks that congest network traffic.

M. E. Ahmed is with the Department of Software, Sungkyunkwan University, Suwon, Republic of Korea, and also with the Data61 of Commonwealth Scientific and Industrial Research Organization (CSIRO), Sydney, Australia (e-mail: ejaz.ahmed@data61.csiro.au).

H. Kim is with the Department of Software, Sungkyunkwan University, Suwon, Republic of Korea. e-mail: (e-mail: hyoung@skku.edu).

S. Ullah is with the Department of Computer Science and Engineering, Kyung Hee University, Yongin, Republic of Korea (e-mail: saeed@khu.ac.kr).

Classifying network traffic applications that generate them can yield suitable measures for countering network attacks, deploying QoS-aware enforcements successfully, and planning for and preventing network congestion. Classifying such applications based on their behavior is a viable approach to combat network attacks, and at the same time predicting legitimate/normal traffic application behavior. Moreover, traffic classification enables network administrators to be aware of the normal behavior of traffic applications and can effectively detect anomalies arising due to various network attacks [4]. Traditional approaches for traffic classification fall into the following categories: 1) Port based, this approach consists of the analysis of the used communication ports. The well-known ports for the protocols are assigned by the IANA. Port based approach is useful only for the applications and services, which use fixed port numbers hence easy to deceive by changing port numbers. As a result, this approach is not valid anymore due to the fact that many applications use dynamic ports (e.g., P2P) and try to disguise themselves by using known port numbers. 2) Payload based techniques that rely on deep packet inspection matches predefined signatures of applications. This technique performs well if the payload is not encrypted, but due to packet's payload encryption and data obfuscation (Skype traffic), these approaches are not effective. Therefore, in order to be aware of the behavior of legitimate traffic applications and at the same time prevent network attacks such as DDoS, we need to effectively classify traffic patterns generated from applications (legitimate or attack).

In our preliminary work, we demonstrated that Software-Defined Network (SDN) can be used to mitigate DNS query-based DDoS attack [18]. We leveraged three flow-level features, such as total number of packets transmitted, ratio of source and destination bytes, and connection duration time, to classify normal and DDoS attack traffic using Bayesian nonparametric clustering. However, in this study, we consider not only flow-level features but also packet-level traffic features to derive a multi-modal probability distribution for generating profiles for normal applications. We also show that those profiles can be used to detect unknown network traffic including attack traffic based on the unsupervised clustering approach.

The main motivation behind our work is that the statistical properties of basic elements of a network flow, such as packet and flow features, can be effectively used to determine which applications are generating legitimate traffic and those that are generating attack traffic. We propose network traffic classification approach which aim to classify traffic patterns based exclusively on their statistical properties. We define the notion of application fingerprints, which take into account packet-level[1] features and generate fingerprints in a compressed but efficient manner. The "known-to-the-system" fingerprinted applications' statistical characteristics are used to generate normal (legitimate) profiles. The "known" application fingerprints are obtained using limited training data, deciding whether a flow belongs to a given application layer protocol or was generated by an "unknown" (i.e., non-fingerprinted) application. The application fingerprints enable us to measure "how far" an unknown traffic flow is from the basic characteristics of each protocol. We utilize flow-level features[2] as well as packet-level features to identify DDoS attacks, and normal network traffic. Here, the "unknown" applications fingerprints, obtained using the proposed traffic clustering approach, are further analyzed to detect whether those clusters are generated from network attacks such as DDoS. The proposed approach is resilient against adversaries emulating legitimate users by adjusting traffic features in order to evade anomaly detection systems. This is owing to the inherent nature of the proposed fingerprinting approach, which takes into account a large number of traffic instances from traffic applications and generates respective application fingerprints, where in practice an adversary can craft fewer traffic instances, and not sufficiently many to generate its own fingerprint to evade the system.

The contributions of this work are summarized as follows:

1) We propose a simple application fingerprinting approach, based on the analysis of simple properties of network traffic such as packet-level features of traffic flows. The fingerprints are obtained in a highly compact and efficient manner. The algorithm for fingerprinting is computationally simple: it involves a histogram-based weighted-mean approach in a window of the observed traffic features. The obtained fingerprints are then used to generate profiles of normal applications based on limited training data.
2) We propose Dirichlet processing mixture model[3] DPMM-based clustering approach to automatically cluster traffic based on applications' flow-level features. DPMM fit well here, because the number of traffic applications is unknown in advance and may increase as more data are observed. DPMM is unsupervised in nature, considering only the observations from traffic, and it is nonparametric, because the number of traffic applications is not known in advance. Unlike previous work in [19], [20], the proposed DPMM generates a multi-modal probability distribution[4] to model the normal traffic profiles of applications. Because traffic applications exhibit unique behaviors [23], it is important to model them using a multi-modal probability distribution rather than a uni-modal one. For example, traffic behaviors under normal and flash events are significantly different [21], and so modeling such traffic using uni-modal statistics is not appropriate.
3) We further propose an approach based on the extended features (packet-level and flow-level) set to detect DDoS attacks (e.g., Slowloris and packet flooding). By flow-level features, we mean the traffic flow's statistics, such as flow duration, total packets/bytes in a flow, and uni- or bi-directional flow characteristics. A DDoS attack shares several packet-level characteristics with normal traffic. Therefore, it is challenging to distinguish attack traffic from normal traffic, and therefore we utilize both flow-level and packet-level features to distinguish network attacks from normal traffic. The proposed detection algorithm decides the legitimacy of an "unknown" application fingerprint based on the traffic flow behavior.

We evaluate the proposed framework through extensive experiments on real network traces from ISCX's intrusion detection dataset [29], CAIDA2007 [30], CAIDA2016 [31], and KAIST [33], [32], [34].

The remainder of this paper is organized as follows. In Section II, we discuss related work concerning attack detection. In Section III, we provide a description of the system model. The definitions of protocol fingerprints and normal traffic profile generation are explained in Section IV. Section V explains the attack traffic detection approach. Experimental results are discussed in Section VI, while Section VII concludes the paper.

II. Related work

In this section, we discuss recent literature on anomaly and DDoS attack detections. We briefly explain their methodologies in order to detect/mitigate network attacks.

Anomaly Detection Recent literature on anomaly detection [4], [5], [7], [8] focuses on network traffic features for attack detection. Crotti et al. [4] exploited the *packet length* and *packet inter-arrival time* for traffic classification using simple statistical fingerprinting. However, although using only packet-level features is effective in traffic classification, this may fail to detect network attacks, because some attacks such as DDoS exhibit similar statistical properties in packet-level features to normal traffic [14], [15], [16]. Therefore, to further improve the classification accuracy, and most importantly to detect attack traffic, relying only on packet-level features may not suffice. In [5], the authors proposed an anomaly detection method which is based on a multi-step outliers. To select the relevant and non-redundant subsets of features for the detection of outliers/anomalies, they leveraged mutual information and generalized entropy based on the selection of features. In [7], Hammamoto et al. proposed a network anomaly detection system using a genetic algorithm and fuzzy logic scheme. They incorporated six traffic flow features, namely bits per second, packets per second, source and destination IP entropies, and source and destination port entropies. Their proposed mechanism is based on an unsupervised learning method. Their experimental results showed that their proposed system achieves a 96.53% accuracy and 0.56% false positive alarms. In [8], Fernandes et al. proposed a profile-based anomaly

[1]Packet-level features refer to per packet details in headers, such as packet length, inter-arrival time, TTL, and IP protocol.

[2]By flow-level features, we mean flow statistics such as flow duration, total packets/bytes, and source/destination bytes.

[3]A Bayesian nonparametric model for clustering.

[4]There is more than one "peak" in the distribution of the data.

detection system. For pattern recognition and anomaly detection, they employed principal component analysis (PCA), ant colony optimization, and dynamic time warping techniques. Their approach generates a traffic profile for normal network behavior, which is compared with the real network traffic to recognize anomalous events. However, we note that under a single normal profile for traffic, the classification performance may be weakened by the dynamic nature of network conditions. For instance, under flash events the profile of normal traffic changes drastically, thereby traditional anomaly detection systems may fail to perform well under such conditions.

DDoS attack detection Typically, attack tools are prebuilt programs, which are usually the same for one botnet [22]. A bot-master initiates an attack command on the compromised hosts (bots) in its botnet to launch an attack session. Evidence for this claim is given in the literature [14], [15], [16]. Consequently, DDoS attack flows received at the targeted web server are an aggregation of many original attack flows, and the aggregated attack flows share a similar standard deviation with the original attack flow.

Detecting DDoS attacks relies on learning normal traffic behavior, so as to classify abnormal behavior of traffic based on the model learned from normal traffic [9]-[12]. In [9], Matta et al. proposed a model for detecting DDoS attacks. The introduced a botnet that emulates normal traffic by continuously learning normal traffic patterns from network traffic. Moreover, they presented an inference algorithm to provide a consistent estimate of possible botnets hidden in the network. In [10], Tan et al. proposed an approach to learning normal traffic behavior, and identifying "known" and "unknown" DoS attacks based on this normal traffic behavior. Their system extracts the geometric correlations between network traffic features using a multivariate correlation analysis for accurate network traffic characterization. Furthermore, they proposed a triangle-area-based Martinique technique to enhance and speed-up the multivariate correlation analysis. Their approach achieved up to a 99.95% accuracy for the KDD Cup 99 dataset. Ben et al. [11] presented an approach to detecting DDoS attacks via identifying malicious users. They employed a vulnerability metric to evaluate the hash table data structure. They further demonstrated that a closed hash is more vulnerable to DDoS attacks than an open hash. One of their key findings is that regular users suffer from a performance degradation even after an attack has ended. In [12], the authors presented an early prediction and prevention method for flooding DDoS attacks. Their proposed mechanism forms a virtual protection ring around hosts to collaborate via exchanging traffic statistics.

In [13], the authors employed fuzzy clustering of TCP packets to detect network intrusions with the reduced set of features. The main motivation behind using fuzzy-based systems is the fact that classification-based systems are often unable to adapt to the dynamic conditions of a network, and moreover they are useful in identifying previously unknown attacks. In [6], the authors experimentally identified that certain common machine learning algorithms under-perform on encrypted malware traffic. One of their key findings is that feature engineering is a decisive factor, and incrementally adding features suggested by domain experts to the initially selected feature set has a positive impact on the performance of a classifier.

III. System model

We briefly present our system for monitoring and detecting normal traffic behavior. In our approach, we progressively utilize traffic features ranging from packet-level to flow-level in order to detect abnormalities in network traffic.

In stealthy DDoS attacks such as Slowloris, packet-level feature-based approaches may allow attack traffic to bypass an anomaly detection system because they are crafted in a way that resembles normal traffic. In order to clearly distinguish the traffic patterns generated by normal and attack traffic, we incorporate flow-level statistics in addition to packet-level features in our method.

A. Packet-level features

The rationale behind packet-level feature inspection is to build profiles of normal traffic applications, and this method is quite effective in classifying traffic applications [4], [18], [**?**], [29]. All traffic features are listed in Table I. We consider the following most important packet-level traffic features for application fingerprinting:

- *Packet length*: Packet lengths for different traffic payloads are likely to be different. For example, the UDP packet size is larger, the HTTP packet size may vary depending on the dynamic behavior of game applications, and VoIP packets have smaller packet lengths in order to minimize jitter. A set of packets with their corresponding lengths is represented as $L_{\mathrm{P}} = \{p_{l_1}, p_{l_2}, \dots, p_{l_N}\}$, where p_{l_n} is the length of the n^{th} packet.
- *Packet inter-arrival time*: Packet inter-arrival times for different applications also vary depending upon the requirements of applications. For example, in VoIP the inter-arrival time is small, in order to avoid undesirable effects caused by jitters. We define the vector of packet inter-arrival times as $I_{\mathrm{P}} = \{p_{i_1}, p_{i_2}, \dots, p_{i_N}\}$, where p_{i_n} is the n^{th} packet inter-arrival time for the traffic application.
- *Time-to-live*: We consider time-to-live (TTL) as a feature, because each traffic application adjusts this value independently. After analyzing various real network traces, we observe that each application sets the TTL value for their packets independently. The TTL value for packets in a flow usually ranges between 40 and 255. We denote this feature as $Ttl_{\mathrm{P}} = \{ttl_1, ttl_2, \dots, ttl_N\}$, where ttl_n is the n^{th} packet's TTL value.

Consequently, the dataset of packet-level features can be represented as

$$\mathbb{X}_p \;=\; [L_{\mathrm{P}},\ I_{\mathrm{P}},\ Ttl_{\mathrm{P}}],$$

where $\mathbb{X}_p$ is matrix containing all features.

Flow-level features are also incorporated in our study due to the fact that certain attacks' flow characteristics show different behavior as compared to the normal traffic application, e.g., flow connection duration, proportion of traffic flow from source to destination and vice versa, etc. We analyzed

TABLE I
REPRESENTATIVE NETWORK FEATURES.

Level	Feature	Description
Packet-level	src/dst IP	source and destination IP addresses
	sr/ds port	source and destination ports
	time_delta	Packet inter-arrival times
	time_relative	Relative start time of a connection
	ip.proto	IP protocol
	ip.hdr_len	IP header length
	ip.len	Packet length
	ip.ttl	Time to live
	tcp.hdr_len	TCP header length
Flow-level	tot bytes	Total flow bytes
	pkts (src $\rightarrow$ dst)	packets from source to destination
	bytes (src $\rightarrow$ dst)	bytes from sources to destination
	pkts (dst $\rightarrow$ src)	packets from destination to source
	bytes (dst $\rightarrow$ src)	bytes from destination to source
	Rel. Start	Relative start time of a connection
	dur.	duration of a connection
	bits (src $\rightarrow$ dst)	bits from source to destination
	bits (dst $\rightarrow$ src)	bits from destination to source

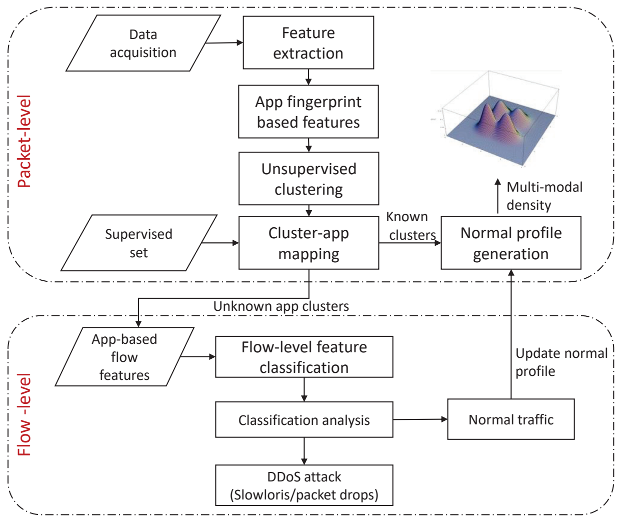

Fig. 1. Framework of the proposed classification scheme for normal profile estimation and attack detection.

the total 18 packet- and flow-level traffic features listed in Table I. However, we selected only those features exhibiting distinguishing behavior among various traffic applications. The selected flow-level features are discussed in Section V.

B. Proposed framework

Fig. 1 illustrates the proposed framework for generating profiles of normal traffic applications and detecting attack traffic based on application fingerprinting. After collecting the data, the proposed framework extracts packet-level features. Packet-level features are used to generate fingerprints for traffic applications. Here, we utilize a histogram-based fingerprinting approach, which is discussed in the following section. Then, the fingerprints (transformed dataset) are clustered using an unsupervised approach, and the output clusters are mapped to their respective application traffic with assistance from the labeled training data. The clusters mapped to applications are called "known" application fingerprints, and those not mapped are termed "unknown" application fingerprints. The known and unknown application fingerprint sets are denoted by K_{fp} and U_{fp}, respectively. The known application clusters are used to generate normal traffic profiles. The generated normal traffic profiles form a multi-modal probability distribution, where each mode in the density function corresponds to a unique traffic application.

As shown in Fig. 1, our approach take into account packet- and flow-level features separately in classifying normal and attack traffic. It is due the following reasons: first, our aim is to quickly generate normal profiles by leveraging the available training data. For that, we utilize only packet-level features and a lightweight clustering algorithm (Mean-shift) to generate normal profiles. Second, by merging packet- and flow-level features and applying classification algorithm on the merged feature may result in clusters which are not specific to characteristics of applications, i.e., traffic from different applications may result in a same cluster. Finally, by analyzing unknown clusters flow-level features separately focuses on detecting attack traffic using Bayesian nonparametric approach (DPMM). Moreover, DPMM may result in new clusters that correspond to various types of DDoS attacks such as Slowloris and packet flooding. In short, packet-level features focuses more on quick generation of normal profiles and flow-level features plays an important role in detecting attacks traffic. As shown in Fig. 1, flow-level features are extracted from the unknown application fingerprinted clusters, and the DPMM-based classification is applied to detect attack and normal traffic.

IV. NORMAL TRAFFIC PROFILES GENERATION USING PACKET-LEVEL FEATURES

In this section, we present our method for generating normal traffic profiles using packet-level features. In practice, the normal behavior of network traffic follow a multi-modal probability distribution [23]. Gu et al. in [19] used the KL divergence between the current and reference traffic to detect anomalies. Scherrer et al. [20] proposed using a unimodal probability distribution, i.e., the gamma distribution, to separate the attack traffic from legitimate traffic. In this study, we rely on a multi-modal probability distribution to model profiles of normal traffic, owing to the fact that real traffic varies significantly according to applications' requirements. Therefore, instead of relying on a unimodal probability distribution, we incorporate a multi-modal distribution in which each mode in the estimated distribution corresponds to a unique traffic application.

A. Application fingerprinting

In this section, we focus on traffic classification based on packet-level statistics produced by network applications exchanging data through TCP connections, such as HTTP, SMTP, DNS, POP, and SSH. Fingerprinting plays an important role and provides summarized information about applications' behaviors in terms of variations in packet-level characteristics. For instance, Fig. 2 illustrates normal and attack traffic behavior in terms of three packet-level features. We clearly see an overlap of among various types of normal traffic in the

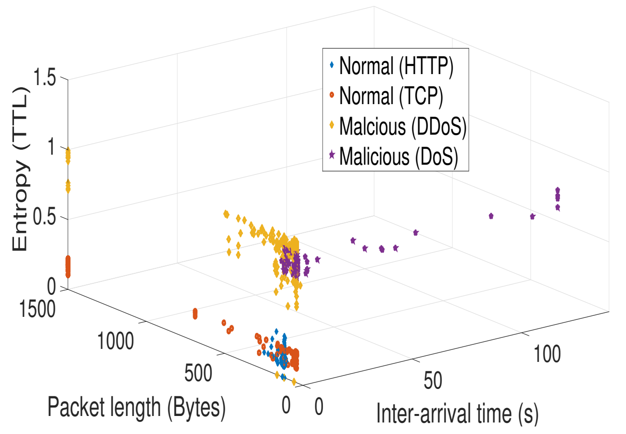

Fig. 2. Packet-level features before application fingerprinting. It is obvious that flooding and Slowloris are located at the same location in the three-dimensional feature space, which makes it difficult for a classifier to classify such attack.

middle-lower part of Fig. 2. Moreover, the two types of DDoS attacks (Slowloris and packet flooding), whose patterns are significantly different from each other, overlap in the middle part of Fig. 2.

The generation of a given application's fingerprint starts from the evaluation of packet-level features. We leverage histogram[5] method on a window of incoming packets at the router's interface to compute weighted-mean (Wm_l) for a set of packets in a window of size W, where the window size W is set according to the proportion of an application's traffic. The weighted mean is similar to an ordinary arithmetic mean[6], except that instead of each of the data points contributing equally to the final average, some data points contribute more than others. The rationale behind using the weighted-mean approach is that packet lengths and their inter-arrival times vary significantly for various applications and are not uniformly distributed across the range of packet lengths and packet inter-arrival times. For instance, in HTTP traffic packet lengths are highly concentrated, either between 50 and 500 bytes or above 1400 (mostly 1500) bytes. Thus, giving equal weights to packet lengths between 50 and 1500 bytes is not suitable. Moreover, for VoIP traffic the packet length is usually 210 bytes, in order to avoid jitters in voice communication [23].

1) Weighted-mean computation of packet inter-arrival time and packet length: First, histogram for a set of incoming packets in a window of size W is computed. The output of a histogram function is a vector of the number of occurrences of random variables $\vec{o}$ and a corresponding vector of the centers $\vec{\mu}$ of the histogram bins, as given in (1). Then, the lth probability density function PDF$_l$ of a histogram is computed using (3), where $L = N/W$ and N is the number of datapoints/observations. Consequently, we obtain total L histogram-based probability density functions PDF$_l$ over N datapoints, as given in (3). We use the number of histogram bins and their corresponding weights to compute L

[5]A histogram is an accurate graphical representation of the distribution of numerical data. It is an estimate of the probability distribution of a continuous variable.

[6]The most common type of arithmetic average.

Algorithm 1 getWeightedMeansVector: Histogram-based PDF and weighted-mean generation.

```
    Input: $\{x_n\}_{n=1}^{N}$, $W$, and $F$.
    where $x_n$ is a packet-level feature, $W$ is the window size, and $F$ is number of bins.
    Result: Weighted-mean vector $\{Wm_l\}_{l=1}^{L}$
1:  Divide feature data in to $L$ disjoint subsets, where $L = \frac{N}{W}$
2:  The $l$th subset, $S_l = \{x_{(n=l)}\}$
3:  for each $l \rightarrow \mathtt{L}$ do
4:      Compute $\vec{o}_{l,b_f}$ and $\vec{\mu}_{l,b_f}$ using Eq. (1)
5:      Compute $\vec{w}_l$ using Eq. (3)
6:      Compute $Wm_l$ using Eq. (2)
7:      $l = l + 1$
8:      $S_l$ = Next window of size $W$ from $\{x_n\}_{n=1}^{N}$
```

PDFs, and corresponding weighted-means $\{Wm_l\}_{l=1}^{L}$ using the following equations:

$$[\vec{o}_{b_f}, \vec{\mu}_{b_f}] = \text{Histogram}(\{x_i\}_{i=1}^{W}, F), \tag{1}$$

$$Wm = \sum_{f=1}^{F} \vec{\mu}_{b_f} Pr(\vec{\mu}_{b_f}), \text{ where} \tag{2}$$

$$Pr(\vec{\mu}_{b_f}) = \frac{\vec{o}_{b_f}}{\sum_{f=1}^{F} \vec{o}_{b_f}}, \tag{3}$$

where F is the total number of bins. Furthermore, $\vec{\mu}_{b_f}$ is a vector containing the centers of F bins, and $\vec{o}_{b_f}$ is the corresponding vector containing the number of occurrences of random variables in each bin. The window sizes used for various traffic applications in our experiments are provided in Table V, where the number of bins F is set to 50 for all applications.

The steps for computing weighted-means are enumerated in Algorithm 1. The input of the algorithm consists of a feature's dataset $\{x_n\}_{n=1}^{N}$, i.e., a packet-level feature's data (packet length L_P, packet inter-arrival time I_P, or time-to-live Ttl_P) and W is the window size. The input feature's data are divided into L windows, each containing W packets (steps 1 and 2 of Algorithm 1). For each $l \in L$, the histogram method is applied to compute the vectors of occurrences $\vec{o}_{l,b_f}$ and bin centers $\vec{\mu}_{l,b_f}$ using Eq. 1 (step 4). Based on $\vec{o}_{b_f}$, the probabilistic weights (PDF) of bins are computed using Eq. 3. Following this, the weighted-mean for the lth PDF is calculated using (2) (steps 5 and 6). This procedure repeats for the $l+1$th subset of feature's data. Algorithm 1 is used to compute the fingerprints for packet lengths and packet inter-arrival times.

2) Weighted-mean computation of TTL: For the packets' TTL features, instead of using the weighted-mean approach for fingerprinting, we employ a window-based entropy method to capture the amount of randomness in TTL values for incoming packets. The rationale behind using entropy lies in the fact that the amount of randomness observed in legitimate applications is diverse [22], whereas in the case of attack traffic the randomness (entropy) in TTL is usually small, or almost zero in some cases. The reason for this is the fact that DDoS programs such as `Mirai` are usually pre-built programs, and almost the same attack pattern is followed by

Algorithm 2 getWindowBasedEntropyVector: Window-based entropy calculation for the packet-level feature TTL.

```
    Input: $\{x_n\}_{n=1}^{N}$ and $W$
    Result: Window-based entropy vector $\{E_l\}_{l=1}^{L}$.
1:  Divide feature data in to $L$ disjoint subsets, where $L = \frac{N}{W}$
2:  The $l$th subset, $S_l = \{x_n\}_{n=1}^{W}$
3:  for all $l \rightarrow$ L do
4:      E$_l$ = Entropy($S_l$)
5:      l = l+1
6:      $S_l$ = Next window of size $W$ from $\{x_n\}_{n=1}^{N}$
```

each infected host. As a result, the traffic statistics generated by all the infected hosts (bots) are deterministic. By exploiting this fact, we aim to incorporate entropy to quantify randomness in TTL for packets. Algorithm 2 presents the steps involved in computing the window-based entropy for Ttl_P. Algorithm 2 is used to compute the fingerprints for packets' TTLs.

3) Applications' fingerprints generation: The application fingerprinting phase of Algorithm 3 enumerate the steps to convert raw dataset to its fingerprinted version, termed as the transformed dataset denoted by $\mathbb{W}$. In the first two steps, weighted-mean vectors of packet inter-arrival times and packet lengths are computed, denoted by $\vec{p}$ and $\vec{t}$, respectively (Step 1 and 2). In third step, entropy for each window is computed, denoted by $\vec{e}$ (Step 3). The transformed dataset $\mathbb{W}$ is obtained by merging the transformed features from the first three step of algorithm into a single matrix form. Note that the dimension of features ($\vec{p}, \vec{t}$, and $\vec{e}$) are identical, therefore, they can be merged to form a fingerprinted ($\mathbb{W}$) features.

Fig. 3 presents applications' packet-level fingerprints following Algorithm 3 (Step 1 to 4). For comparison, we also show the non-fingerprinted version of the data for the same features in Fig. 2. We suspect that classification algorithms would suffer from performance issues while classifying packet-level features shown in Fig. 2, whereas the same classifier would perform optimally on the finger-printed features shown in Fig. 3. Moreover, we can clearly see that respective types of normal traffic are finely separated in the three-dimensional feature space. Similarly, different attack types are also finely separated in the feature space.

B. Mean-Shift clustering

Given $\mathbb{W} = \{\vec{p}, \vec{t}, \vec{e}\}$, we employ the mean-shift (MS) clustering approach to cluster the transformed dataset $\mathbb{W}$. MS is a non-parametric feature-space analysis technique for locating the maxima, i.e., the modes, of a density function given a dataset, and can also be called a mode-seeking algorithm [25]. This is an iterative approach, and starts with an initial estimate x, where a kernel function $K(x_i - x)$ is given. This function determines the weights of nearby points for the re-estimation of the mean. Typically, a Gaussian kernel on the distance to the current estimate is adopted, i.e., $K(x_i - x) = \exp^{-c||x_i - x||^2}$. The MS clustering aims to partition the traffic flows into k clusters, i.e., $C = \{C_1, C_2 \dots, C_k\}$, where each cluster has its own respective cluster parameters.

The transformed dataset $\mathbb{W}$ is input to the unsupervised MS clustering algorithm, which assigns each datapoint a cluster

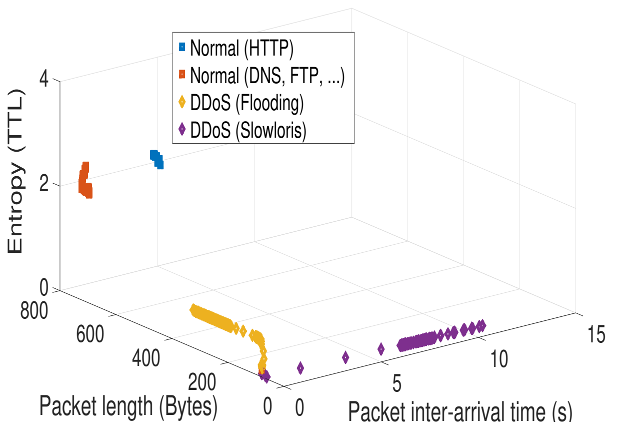

Fig. 3. Packet-level features after application fingerprinting. It is evident from the figure that normal traffic types (HTTP, FTP, and DNS) are clearly separated in the three-dimensional feature space. The DDoS traffic types (unknown fingerprints) are also clearly separated, but we require further analysis to avoid false alarms from classifiers, because we still see some normal traffic in DDoS traffic (red squares).

assignment variable $\{c_n\}_{n=1}^{K}$, where $K < N$. The cluster assignment variable c_n indicates the cluster to which that datapoint belongs to. The datapoints from the same application will have same cluster assignment variable (c_n). MS clustering will result into K clusters with their respective cluster parameters. Note that the numbers of clusters obtained in each case, i.e., for normal and attack traffic, are expected to be greater than one. The reason for this is that for legitimate applications, the distribution is assumed to be multi-modal. Similarly, for attack traffic, if there is more than one attack then one cluster is created for each attack type.

C. Normal profile generation

Profiles for normal applications are generated based on training data. Algorithm 3 enumerates the step for generating normal profiles for applications. First, application fingerprints are obtained (Step 1 - 4). The obtained fingerprinted features $\mathbb{W}$ are input to the MS clustering algorithm (Step 5). We assume that limited training data are available for normal traffic applications such as HTTP, FTP, and DNS. Using the training data, the K clusters obtained from MS clustering are checked in-turn for their similarity with the training data. Based on the similarity scores, the clusters are mapped to applications. Next, clusters are categorized into two groups: known fingerprinted cluster set (K_{fp}) and unknown fingerprinted clusters (U_{fp}). The clusters in K_{fp} are used to generate the profiles of normal applications (Step 6 - 9), whereas those in U_{fp} are further analyzed to confirm the legitimacy of the applications' fingerprints.

Note that our dataset consists of both legitimate (various benign applications) and DDoS attack types (Slowloris and flooding) traffic. The clusters for which mapping is found comes in K_{fp} set of clusters. Whereas, those clusters which are not known may contain legitimate and/or attack traffic, and hence comes under the set of unknown clusters U_{fp}, which needs further analysis. The clusters in U_{fp} may consist of legitimate and/or attack traffic. After analyzing the clusters in U_{fp}, the decision about their legitimacy are made.

V. Flow-level feature-based attack traffic detection

In this section, the unknown fingerprints U_{fp} are further analyzed using flow-level features in addition to the packet-level fingerprints, to decide whether the data of the fingerprinted clusters have been generated by a legitimate or attack application. The reason for this further analysis the fact that certain network attacks, such as DDoS, exhibit similar characteristics in packet-level features to normal traffic [17], [21]. For instance, the behavior of flash events is quite similar to DDoS behavior in terms of packet-level features. However, because DDoS attacks are controlled by a botmaster, the correlation between DDoS flows is very high [21], [22]. Thus, by exploiting this fact we propose a flow-level features-based approach to identifying attacking and normal applications.

A. Flow-level feature extraction from unknown fingerprints

In this subsection, we investigate the flow-level features to further confirm the legitimacy of the unknown fingerprinted clusters set U_{fp}. Here, the flow features from U_{fp} are extracted. The flow-level features provide a holistic view of traffic flow behavior.

We consider the following extended packet- and flow-level features:

1) Flow-duration (F_d): The flow duration plays an important role in detecting normal and attack traffic [18]. Therefore, we consider this for our proposed method. It is represented here as $F_d = \{fd_1, \dots, fd_N\}$, where N is the number of flows.
2) Ratio of source and destination bytes (R_f): The ratio of packets between the source and destination hosts can be used to detect the normal activity pattern between the host and the web server [18]. We denote this as $R_f = \{fr_1, \dots, fr_N\}$.
3) Packet-level fingerprinted features: Here, we reuse the packet-level fingerprinted features ($\vec{p}$) obtained using Algorithm 1. The inter-arrival time distribution plays an important role in detecting misbehaving applications. For example, Slowloris is an example of a DDoS attack that keeps the connection busy for a long time and makes the server perform computationally intensive tasks in order to bring the web server down. This feature is denoted as $\vec{p} = \{fp_1, \dots, fp_N\}$.
4) Packet-level fingerprinted features: Similarly, we reuse another fingerprinted feature ($\vec{t}$) to quantify the behavior of traffic flows in terms of randomness in the TTL of packets. This is represented as $\vec{t} = \{fe_1, \dots, fe_N\}$.

The dataset for the extended traffic features can be represented as

$$\mathbb{Y} = \{y_1, \cdots, y_N\}, \quad (4)$$
$$\text{where } y_n = \{fd_n, fr_n, fp_n, fe_n\}. \quad (5)$$

Algorithm 3 Application fingerprint-based normal profile generation.

```
Input: $\mathbb{X}_p = \{x_n\}_{n=1}^N$
Result: $K_{fp}$, $U_{fp}$

Application fingerprinting:
1: $\vec{p}$=getWeightedMeansVector($L_P$) using Algo. 1
2: $\vec{t}$=getWeightedMeansVector($I_P$) using Algo. 1
3: $\vec{e}$=getWindowBasedEntropyVector($Ttl_P$) using Algo. 2
4: $\mathbb{W} = \{\vec{p}, \vec{t}, \vec{e}\}$

Clustering:
   /*Apply MS clustering */
5: $\{c_1, \dots, c_N\}$ = MS-Clustering($\mathbb{W}$)
   /* $K$ clusters (both legitimate and attack) are obtained with respective cluster parameters. */

Normal profiles generation:
6: for all $k \to K$ do
7:     if the training data represent the $k$th cluster $C_k$ then add $C_k$ cluster to $K_{fp}$.
8:     else add $C_k$ to $U_{fp}$.
9: Generate normal profiles for legitimate traffic using $K_{fp}$.
10: return $U_{fp}$ for further processing.
```

Based on the features set $\mathbb{Y}$, we intend to cluster traffic flows to detect attack and legitimate applications. Let us take a real-world example to explain the motivation behind incorporating flow-level features. Consider two flows (one legitimate and the other an attack) $\mathbf{f}_a$ and $\mathbf{f}_b$, initiated by two different hosts targeting the same destination host at TCP port 80. If packet-level features are analyzed, such as the packet inter-arrival I_P time and packet lengths L_P for the two flows, they may exhibit similar behavior, and the anomaly detector may let both flows pass. However, if the flow duration F_d and the ratio of source to destination bytes R_f are analyzed, the normal flows would exhibit a significant variation compared to the attack flow. For instance, in the case of a DDoS attack, the variation in TTL values is negligible, because they are controlled by a single built-in program, whereas the normal traffic shows significant variations in TTL values, because it is controlled by different applications, and each application sets these values according to their requirements. Similarly, packet inter-arrival times are expected to be higher and less random in the case of a Slowloris attack.

B. DPMM-based clustering

Here, the extended feature set $\mathbb{Y}$ is input into the proposed DPMM-based clustering method to identify legitimate/attack applications. Rather than relying on heuristics (e.g., as shown in Fig. 7, TTL values could be used to detect attack traffic) for attack application detection, we consider DPMM-based clustering for detecting different types of normal and attack traffic applications. The DPMM models are flexible in detecting unique patterns in a dataset using clustering. Our goal here is to detect attacks of different types. Of course, in this study we only restricted ourselves to Slowloris and flooding attacks, but in future other types of attack may show-up forming a new cluster. Under such situations, approaches relying on heuristics may fail. DPMM which comes under the umbrella of Bayesian nonparametric statistics are flexible in allowing the formation of new clusters as more data are observed. Thus, unknown attack types of attacks or normal traffic applications could be effectively detected using DPMM.

In practice, we do not know how many traffic applications (legitimate and attack) are active in a network, and therefore we would like to determine this from the observed data. Dirichlet processes can be employed to obtain a mixture model (DPMM) with infinite components (applications), which can be viewed as taking the limit of the finite mixture model for $K \rightarrow \infty$. In the following, we briefly discuss the DPMM framework which is unsupervised and nonparametric model for clustering. However, for detailed overview about Dirichlet processes and respective models, please refer to [26], [27], [35], [36].

A graphical model of the DPMM is presented in Fig. 4. Here, α is the scalar hyperparameter of the DPMM, which affects the number of clusters obtained, and Z_n is the cluster assignment variable such that the feature point y_n belongs to the kth cluster. The larger the value of α, the more clusters, and the smaller the value of α the fewer clusters. Note that the value of α indicates the strength of belief in G_o, where G_o is the base distribution. A large value means that most of the samples will be distinct, and have values concentrated around G_o. In addition, θ_k are the cluster parameters, where $k \in \{1, 2, \dots\}$. The mixture distribution can be defined as $p(y_n) = \sum_{k=1}^{\infty} \pi_k \; p(\cdot|\delta_{\theta_k^*})$, where the mixing weights are denoted by π_k and the mixing components (clusters) are represented by $p(\cdot|\delta_{\theta_k^*})$. Here, $\delta_{\theta_k^*}$ is employed as compact notation for $\delta(\theta = \theta_k^*)$, which is a delta function that takes a value of 1 if $\theta = \theta_k^*$ and 0 otherwise.

We represent the generative model for the DPMM using the stick-breaking process [35]. Consider two infinite collections of independent random variables, $V_k \overset{i.i.d.}{\sim} \text{Beta}(1, \alpha)$ and $\theta_k^* \overset{i.i.d.}{\sim} G_o$, $k = \{1, 2, \dots\}$. The stick-breaking process of G is given by

$$\pi_k = V_k \prod_{j=1}^{k-1} (1 - V_j), \tag{6}$$

$$G = \sum_{k=1}^{\infty} \pi_k \delta_{\theta_k^*}, \tag{7}$$

where $\mathbf{V} = \{V_1, V_2, \dots\}$ with $V_0 = 0$. The mixing weights $\{\pi_k\}$ are given by breaking a stick of unit length into infinitely small segments. In the DPMM, the vector $\boldsymbol{\pi}$ represents the infinite vector of mixing weights, and $\{\theta_1^*, \theta_2^*, \dots\}$ are the atoms, which correspond to mixing components. Because Z_n is the cluster assignment random variable for the feature point y_l, the data for the DPMM are generated as follows:

1) Draw $\pi_k|\alpha \overset{i.i.d.}{\sim} \text{Beta}(1, \alpha)$, $\quad k \in \{1, 2, \dots\}$.
2) Draw $\theta_k^* \overset{i.i.d.}{\sim} G_0$, $\quad k \in \{1, 2, \dots\}$.
3) For the feature point y_n, do:
   (a) Draw $Z_n|\{v_1, v_2, \dots\} \sim \text{Multinomial}(\boldsymbol{\pi})$.
   (b) Draw $y_n|z_n \sim p(y_n|\theta_{z_n}^*)$.

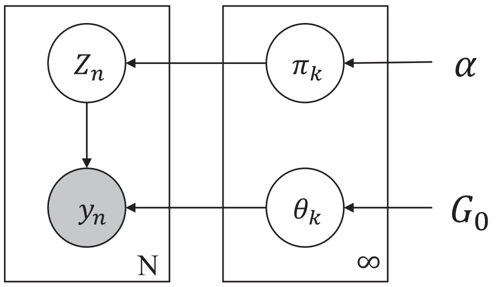

Fig. 4. Graphical model of the DPMM. Nodes represent random variables, edges are dependencies, and plates are replications. The shaded node represents the observations.

Algorithm 4 Extended traffic feature-based normal and attack flow detection.

```
   Input: $\mathbb{Y} = \{y_n\}_{n=1}^N$, $Ent_{\text{thr}}$, and $I_{\text{thr}}$
   Result: Attack detection

   DPMM clustering:
1: $\{C_1, \dots, C_K\}$ = DPMM-Clustering($\mathbb{Y}$)

   Attack and normal traffic detection:
2: for all $k \rightarrow \mathtt{K}$ do
3:     $C_{k,\vec{t}}$ = Avg($C_k(\vec{t})$)
4:     $C_{k,R_f}$ = Avg($C_k(R_f)$)
5:     $C_{k,\vec{p}}$ = Avg($C_k(\vec{p})$)
6:     if $C_{k,\vec{t}} < Ent_{\text{thr}}$ and $C_{k,R_f} = 0$ then
7:         if $C_{k,\vec{p}} > I_{\text{thr}}$ then Label $C_k$ as Slowloris attack.
8:         else Label $C_k$ as flooding attack.
9:     else Label $C_k$ normal traffic application and update
       the corresponding normal traffic profiles.
```

Here, we restrict ourselves to a DPMM for which the observable data are drawn from a normal distribution, and where the base distribution for the DP is the corresponding conjugate distribution.

Because Dirichlet processes are nonparametric, we cannot use the EM algorithm to infer the random variables $\{Z_n\}$ (which store the cluster assignments) for the proposed DPMM model, owing to the fact that EM is generally used for inferences in a mixture model, but here G is nonparametric, making the application of EM difficult. Hence, in order to estimate these assignment variables in the paradigm of Bayesian nonparametrics, a sampling-based approach based on Markov chain Monte Carlo (MCMC) is leveraged to cluster $\mathbb{Y}$.

We propose using the collapsed Gibbs sampling approach [26], which requires selecting a base distribution G_o that is a conjugate prior of the generative distribution $p(y_n|\theta_{z_n}^*)$ in order to analytically solve and sample directly from $p(Z_n|Z_{-n}, \mathbb{Y})$. The posterior distribution under our model is

$$P(Z, \Theta, \boldsymbol{\pi}, \alpha) \propto P(\mathbb{Y}|Z, \Theta) P(\Theta|G_0) \prod_{n=1}^{N} P(z_n|\boldsymbol{\pi}) P(\boldsymbol{\pi}|\alpha) P(\alpha) \tag{8}$$

TABLE II
DATASETS

	Dataset	Protocol	Protocol composition	Size (GB)	Description
Attack traffic	ISCX [29]	TCP, UDP	HTTP, FTP, DNS	84	DDoS (Slowloris and flooding), and Normal traffic.
	CAIDA DDoS[30]	TCP	HTTP,SMTP,SSH	9.27	DDoS attack traffic
Normal traffic	CAIDA Normal [31]	TCP	HTTP, FTP	21	Normal traffic.
	KAIST/WiBroSource [32]	UDP	VoIP, CBR	4	UDP protocol-based CBR and VoIP traffic.
	SNU/WoW [33]	TCP	War of Warriors game	0.2	Online game traffic using TCP protocol.
	SNU/bittorrent [34]	TCP	P2P	0.8	Bittorrent traffic.

By integrating-out certain parameters, the posterior distribution is given by [26] as

$$P(z_n = k|Z_{-n}, \mathbb{Y}, \Theta, \boldsymbol{\pi}, \alpha) \propto P(y_n|z_n, \Theta)P(z_n|Z_{-n}, \alpha) \tag{9}$$

For the first term in the above equation, we use the multivariate Student-t distribution, i.e., $t_{\nu_n-2}[\vec{\mu}_n, \frac{\Lambda_n(\kappa_n+1)}{\kappa_n(\nu_n-2)}]$, because we chose the inverse-Wishart as a conjugate prior for Σ_k and the normal distribution for $\vec{\mu}_k$, where the second term is called a Chinese restaurant process, and is given by

$$P(z_n = k|Z_{-n}) = \begin{cases} \dfrac{m_k}{n-1+\alpha}, & \text{if } k \leq K_+, \\ \dfrac{\alpha}{n-1+\alpha}, & \text{if } k > K_+ \end{cases} \tag{10}$$

where $Z_{-n} = Z/z_n$, K^+ is the number of classes containing at least one data point, and $m_k = \sum_{n=1}^{N} I(z_i = k)$ is the number of data points in the class k. The steps involved in collapsed Gibbs sampling are enumerated below:

1) Initialize the cluster assignments $\{z_n\}$ randomly.
2) Repeat the following until convergence:
   (a) Randomly select y_n.
   (b) Fix all other z_n for every $n \neq n$: Z_{-n}.
   (c) Sample $z_n \sim p(Z_n|Z_{-n}, \mathbb{Y})$ from (10).
   (d) If $z_n > K$ then update $K = K + 1$.

After convergence, the cluster labels Z_n are assigned and the cluster parameters are computed.

C. Clustering-based normal/attack traffic identification

Algorithm 4 presents the steps required to further analyze flows and detect whether they are normal or attack flows. The algorithm takes the feature set $\mathbb{Y}$, unknown fingerprinted clusters set from Algorithm 3, the threshold for randomness in flows' TTL values, and the threshold for the flows' packet inter-arrival times as input.

Thresholds (Ent_{thr} and I_{thr}) and the ratio of source to destination bytes (C_{k,R_f}) play an important role in detecting certain types of attacks. Since we have attack dataset for DDoS, we compute thresholds from clusters formed by the attack traffic. We observe that most benign applications such as VoIP, P2P, Game, TCP/UDP show reasonable variations in entropy in TTL (all applications' entropy in TTL are greater than 1). Following this observation, we set the threshold the $Ent_{\text{thr}} = 1.2$. Similarly, due to the inherent nature of the Slowloris DDoS attack, the inter-arrival time between packets are higher than that of benign applications, as shown in Fig. 7. Of course, traffic inter-arrival time for various benign traffic applications could be different ranging from micro- to milli-seconds, however, for the Slowloris attack, the inter-arrival times mostly lies above 4 seconds which is significantly greater than all benign applications. Due to this, we set the threshold $I_{\text{thr}} = 4$ following the cluster parameters of benign and attack traffic.

TABLE III
ISCX DATASET DESCRIPTION.

Scenario	Activity	Size (GB)
S1	Normal activity only	16.1
S2	Normal	4.22
S3	Network infiltration from inside + Normal	3.95
S4	HTTP Denial of Service + Normal Activity	6.85
S5	DDoS using an IRC Botnet + Normal Activity	23.4
S6	Normal Activity + bruteforce attacks	17.6
S7	Brute Force SSH + Normal Activity	12.3
	Total =	84.4

First, the DPMM-based clustering algorithm is applied to obtain clusters $\{c_n\}_{n=1}^{N}$. The clusters obtained from DPMM are analyzed to classify them as attack or normal traffic applications. Then, for each cluster the averages of the following three features are computed: entropy in TTL values, ratio of source to destination bytes, and flow packet inter-arrival time (steps 3-5). If a cluster's entropy in TTL is less than the threshold and the ratio of source to destination bytes is zero, this means that the referenced cluster likely belongs to attack traffic. The attack types are checked in step 7, i.e., to detect whether the attack type is Slowloris or flooding (steps 6–9). If a cluster is labeled as normal traffic, then the normal profile generated in Section V is updated accordingly.

VI. EVALUATION

In this section, we perform extensive experiments to comprehensively evaluate the proposed approach for normal profile generation and attack traffic identification. To evaluate the proposed approach for application classification, we manually set some identified applications as unknown in the experiments.

A. Datasets

In this study, we evaluate the performance of the proposed approach using real network traffic measurements, the ISCX intrusion detection evaluation dataset [29], CAIDA2007 [30], and CAIDA2016 [31]. The ISCX dataset consists of normal and attack traffic. The normal dataset traces consist of real traffic for HTTP, SMTP, SSH, IMAP, POP3, and FTP. Each trace is exactly 24 hours long. The total ISCX dataset consists

TABLE IV
SUMMARY OF THE MEASURED TRAFFIC DATA.

ISCX 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

TABLE V
FINGERPRINTING PARAMETERS

Traffic (CAIDA and ISCX)	Instances	Window size
Normal HTTP (CAIDA)	1,834	10^2
Normal (TCP+ HTTP + FTP+ DNS)	158,761	10^3
DDoS by IRC botnet	138,850	10^3
HTTP based Slowloris	9,211	10^2
VoIP (UDP-based)	299,003	10^3
WoW game	17,577	10^2
P2P	521,688	10^3
CBR (UDP-based)	898,294	10^3

of 84.4 GB of data, comprising different scenarios. Each scenario, along with the respective traffic size, is listed in Table III. In our experiments, we utilized the scenarios S1, S4, and S5 to evaluate the proposed approach.

The CAIDA2007 [30] dataset contains approximately one hour of anonymized traffic traces from a DDoS attack on August 4, 2007. This type of DDoS attack attempts to block access to the targeted server by consuming computing resources on the server and consuming all of the bandwidth of the network connecting the server to the Internet. The total size of the dataset is 21 GB. These traffic traces recorded only attack traffic to the target and responses from the target, whereas the legitimate traffic has been removed as far as possible. The CAIDA2016 dataset [31] consists of anonymized passive traffic measurements from passive monitors in 2016. It contains traffic traces from the 'equinix-chicago' high-speed monitor. Normal traffic datasets of CBR and VoIP traffic measurements are from the WiBro network in Seoul, Korea [33]. Similarly, normal traffic application behavior of P2P and the War-of-Warriors (WoW) game are obtained from [32] and [34], respectively.

For multi-modal normal profile generation, we utilize data from both the CAIDA2016 and ISCX normal activity scenarios (S1 and S2 of Table III). For attack traffic datasets, we use HTTP-based DDoS (S4 from Table III) and IRC-based DDoS attack data (S5 from Table III) from ISCX. We also use the DDoS dataset from the CAIDA2007 dataset. For the S4 and S5 scenarios in Table III, we consider the traffic from malicious hosts targeting the server, whereas for normal traffic we utilized traffic from all hosts to the targeted server. Finally, normal and attack datasets are mixed together to evaluate the proposed approach for traffic classification-based normal and attack traffic detection.

B. Implementation considerations

The traffic composition of the ISCX traffic dataset is shown in Table IV. The dataset consists of three classes (HTTP: about 87%, FTP: 6%, attack: 7%). We conducted the experiments on a dual core i7-4790 CPU with a speed of 3.60 GHz in each core and installed a memory of 16 GB.

1) Fingerprinting parameter setting: Because we employ a histogram method for fingerprinting applications, we need to set the number of bins F and window sizes W. We set $F = 50$ for all experiments. The reason for fixing this for all experiments is that we analyzed from real traces that $F = 50$ can fully represent the variations in packet-level features. However, we set W according to each application's proportion, i.e., for different applications W is set depending on the size of that application with respect to other applications in the whole dataset. Table V presents the number of instances of each application with the respective window sizes used in our experiments.

2) Hyperparameter setting for DPMM clustering: In our simulations, we follow the generative model in (6) and (7) to generate data with initial parameters from hyperparameters G_0. In DPMM, we need to set the hyperparameters α and G_0 $=\{ \vec{\mu}_0, \kappa_0, \Lambda_0^{-1}, \nu_0\}$, as shown in Fig. 4. Here, α encodes the number of clusters, as implied by (10). When α is larger, the probability of assigning an observation to an unrepresented class is greater than for a represented class. Hence, the value of α is set according to the belief in the number of clusters in the dataset. In our experiments, we set this to 1. For the generative model hyperparameters G_0, we need to set the corresponding conjugate hyperparameters of a base distribution [36]. We see that the mean vector of a class follows the Gaussian distribution with mean $\vec{\mu}_0$ and covariance matrix Σ_k/κ_0. Thus, the two required hyperparameters are $\vec{\mu}_0$ and κ_0, which reflect the mean locations of clusters. If κ_0 is greater, then the clusters are located close to each other, and conversely if κ_0 is smaller, then the clusters will have a larger variation in Σ_k/κ_0, and hence the clusters are sparser. Consequently, $\vec{\mu}_0$ and κ_0 are selected accordingly based on the prior knowledge about the observations. The hyperparameters of the inverse-Wishart distribution, which is the conjugate prior of the multivariate normal, are Λ_0^{-1} and ν_0. Here, Λ_0^{-1} is the scale matrix for the distribution, and ν_0 is the degree of freedom. The hyperparameters are set to $G_0 = \{\Lambda_0^{-1}, \nu_0, \vec{\mu}_0, \kappa_0\} = \{\text{Identity}(3), 4, \text{Zeros}(3,1), 0.5\}$. For details, we refer the reader to [36].

C. Application fingerprinting analysis

Here, we demonstrate the efficiency of the proposed application fingerprinting method, and the corresponding classification results on the fingerprints. Unlike directly applying a classification algorithm on the non-fingerprinted packet-level features, which may result in a higher misclassification rate, we first take the packet-level features and perform the proposed fingerprinting method to obtain applications' fingerprints, and then these fingerprints are input into the classifica-

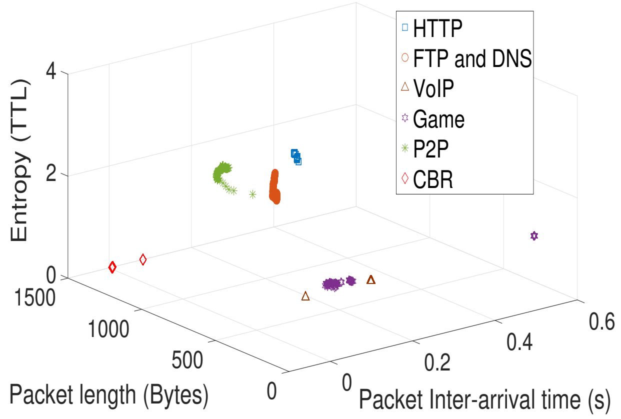

Fig. 5. Fingerprints for various traffic applications.

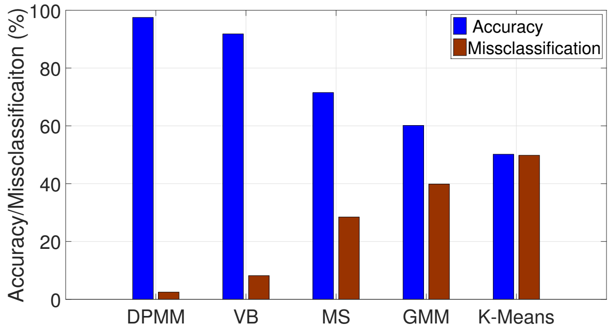

Fig. 6. Traffic application identification accuracy/misclassification rate for normal and attack traffic for the proposed DPMM approach compared with variational Bayes (VB), Gaussian mixture model (GMM), mean-shift (MS), and K-means.

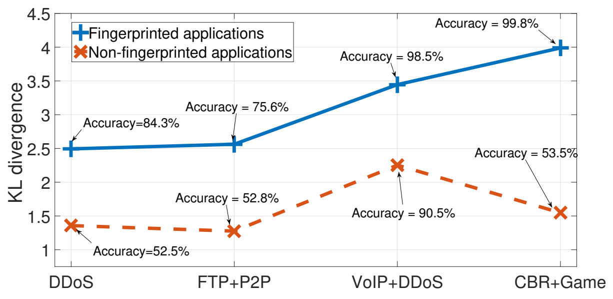

Fig. 7. KL divergence and accuracy for proposed application fingerprinting and non-fingerprinting of applications.

tion algorithm (MS) in order for it to perform optimally with minimal misclassifications.

Fig. 5 shows the fingerprints generated for various traffic applications. Note that the fingerprints are finely separated in the three-dimensional feature space, which leads to optimal classification results. It is observed that HTTP and gam3 (WoW) traffic show significant variations in their traffic patterns. For instance, in Fig. 5 there seem to be two clusters for the WoW game (purple-stars) traffic. This variation in the game traffic is due to game traffic dynamics: it may sometimes produce misclassifications when input to a classifier.

We leverage the Kullback–Leibler (KL) divergence to quantify the distances among various fingerprints generated from applications. To quantify the efficiency of our fingerprinting method, we compute the KL divergence for both the fingerprinted and the non-fingerprinted application data. The KLD is a measure of how one probability distribution, such as a traffic application, diverges from a second. The KL divergence for the multivariate case (packet-level features) is represented as follows:

$$\mathrm{KLD} = \frac{1}{2}(\mathrm{trace}(\Sigma_2^{-1}\Sigma_1) + (\vec{\mu}_2 - \vec{\mu}_1)^T \Sigma_2^{-1}(\vec{\mu}_2 - \vec{\mu}_1) - \ln(\frac{det\Sigma_1}{det\Sigma_2} - D)),$$

where Σ is a covariance matrix for a cluster with D features, and $\vec{\mu}$ represents the cluster mean. Fig. 7 shows KL divergences and the corresponding classification accuracies for both the fingerprinted and non-fingerprinted applications. The larger the KL divergence between two distributions (clusters), the easier it is to separate them. From Fig. 7, the KL divergence for fingerprinted applications is seen to be higher than for those that are non-fingerprinted. The corresponding classification accuracy against each KL divergence for pairs of clusters (applications) is shown to be significantly higher than for the proposed fingerprinting method. We observe a classification accuracy of up to 99.8% for CBR and WoW traffic. This is because of the fact that the KL divergence (separation between two clusters) is large between these. For further verification, we observe that the distance between the CBR (red-diamond) and the WoW (purple-star) clusters in Fig. 5 is significantly large (around 4), which results in a higher classification accuracy (99.8%). However, the KL divergence between the FTP (red-circle) and the P2P (green-star) clusters is comparatively low (2.51), owing to which the classification accuracy is also low (75.6%), but it is still significantly higher than in the corresponding non-fingerprinted case (52.8%).

D. Flow-level feature-based normal and attack traffic identification

In this subsection, we present a performance evaluation of the proposed method in terms of correctly identifying attack and normal traffic applications. It should be noted that the core of Algorithm 4 is DPMM clustering. Therefore, we present here the flow-level traffic identification accuracy for DPMM, and then we compare the performance with four classification approaches. For comparison, we take variational Bayes inference (VB), mean-shift (MS), Gaussian mixture model clustering (GMM), and K-means. The experiments performed in this subsection consist of a dataset containing 2,198 data-points including normal and attack traffic. There are total eight traffic applications, i.e., six normal and two attack traffic.

The variational Bayes (VB) inference approach is based on variational methods, which convert inference problems into optimization problems [35]. The main idea that governs variational inference is that it formulates the computation of a marginal or conditional probability in terms of an optimization problem, which depends on the number of free parameters, i.e., variational parameters. We use the same experimental

TABLE VI
ATTACK AND NORMAL TRAFFIC APPLICATION IDENTIFICATION RESULTS.

Algorithms	Normal traffic	DDoS (flooding using IRC botnet)	DDoS (Slowloris)	Overall accuracy	Misclassification rate
Proposed DPMM	100%	99.2%	93.4%	97.5%	2.5%
Variational Bayes Inference	62.7%	100%	95.6%	91.8%	8.2%
Mean-Shift	9.8%	82.6%	89.1%	71.5%	28.5%
Gaussian Mixture clustering	60.7%	100%	0%	60.1%	39.9%
K-means	54.9%	81.8%	0%	50.1%	49.9%

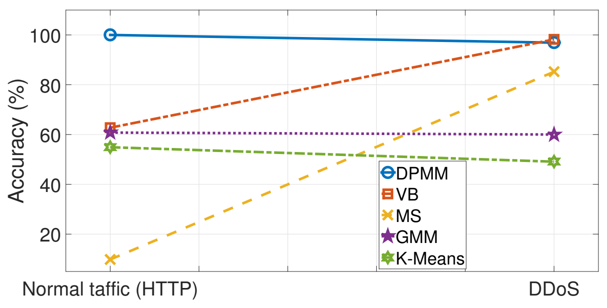

Fig. 8. Normal and aggregated attack traffic identification of the proposed DPMM approach, and the VB, MS, GMM, and K-means approaches.

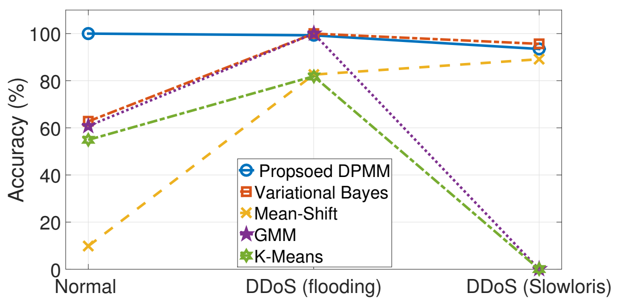

Fig. 9. Traffic application identification accuracy comparison for variational Bayes (VB), Gaussian mixture model (GMM), mean-shift (MS), and K-means.

setting used in [35]. The prior knowledge for the number of clusters in set to 10, i.e., $K = 10$. The $\alpha = 1$ which states that the clusters are equiprobable. All the distributions are assumed to be Gaussian in optimization-based variational inference. Here, the variational parameters give the marginal or conditional probabilities of interest. Unlike the proposed DPMM approach, which is based on a Markov-based sampling approach, the VB approach is based on an optimization approach.

Gaussian mixture model (GMM)-based clustering is a probabilistic model for representing normally distributed clusters within the overall data [27]. In general, mixture models do not require knowledge of which cluster a data point belongs to, allowing the model to learn the cluster centers automatically. Because clustering assignments are not known, this constitutes a form of unsupervised learning. Since GMM-based clustering is parametric in nature, i.e, the number of clusters need to be provided as prior, therefore, we set the number of cluster to $K = 8$. This dictates that the classifier should partition the provided dataset into 8 clusters. Even though, we provide the number of clusters in advance, Still GMM underperforms as compared to the DPMM.

Finally, the K-means algorithm is a well-known clustering approach, which takes the dataset and number of clusters K as input and partitions the input dataset into K clusters, while each cluster has its own cluster center [24]. Similar to GMM-based clustering, we have set the number of clusters to $K = 8$ while performing experiments because K-means is also a parametric clustering approach.

Fig. 6 presents the overall traffic application identification accuracies and misclassifications. By the clustering accuracy, we mean the overall accuracy in correctly detecting normal (HTTP, FTP, and DNS) and DDoS attack application traffic. In other words, this refers to the total number of correctly identified data points over the total number of data points. From Fig. 6, we observe that the proposed DPMM traffic clustering accuracy is significantly higher 97.5% than those of the other approaches. However, the accuracy of the VB clustering approach lies at around 91.8%, whereas all the other approaches struggle to correctly detect the traffic applications. The MS, GMM, and K-means approaches suffer from low detection rates and higher misclassifications. The misclassification rates for all of these approaches lie below 60%. Overall, the proposed DPMM approach performs significantly better in detecting normal traffic. It is evident from Fig. 6 that the proposed DPMM misclassification rate is around 2.5% for all traffic applications, whereas for VB it is around 8.2%. Table VI provides comprehensive results regarding the accuracy for various traffic applications.

Fig. 9 shows the per-application accuracy results for various algorithms. It is evident that the proposed DPMM-based approach accurately identifies normal traffic (HTTP, FTP, and DNS) with 100% accuracy. However, the attack traffic is classified with very few misclassifications. Overall, the proposed DPMM outperforms all other algorithms, with 97.5% accuracy results. Note that the VB approach struggles to identify normal traffic applications. However, GMM and K-means perform worse in detecting Slowloris attacks. The MS, GMM, and K-means approaches perform better in identifying DDoS attacks, but at the same time traffic from Slowloris is mislabeled as flooding attack traffic.

Fig. 10 illustrates attack-only traffic detection accuracies. We see that both the DPMM and VB approaches perform well in detecting attack traffic. Even VB performs well in detecting Slowloris attacks compared to the proposed DPMM. However, VB struggles to detect normal traffic applications and mislabels them as attack traffic. Owing to this, the overall accuracy of DPMM is significantly better than that of the VB approach. From Fig. 10, GMM and K-means perform worse

TABLE VII
NORMAL TRAFFIC APPLICATION IDENTIFICATION ACCURACY.

Algorithms	HTTP	VoIP	WoW game	P2P	CBR	Overall acc.
Proposed DPMM	98.1%	100%	71.6%	99.8%	100%	96.5%
Variational Bayes	0%	0%	100%	99.8%	100%	82.4%
Mean-Shift	100%	100%	0%	99.4%	100%	87.9%
Gaussian Mixture	100%	100%	0%	99.6%	100%	88.1%
K-means	100%	100%	100%	40.7%	100%	84.7%

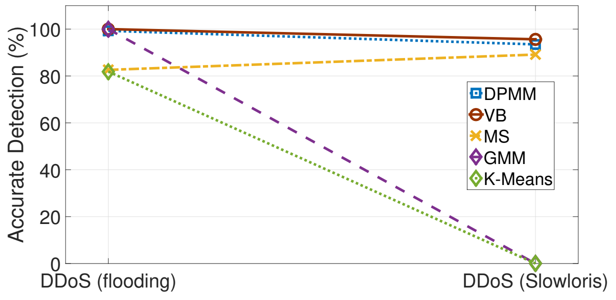

Fig. 10. Attack-only traffic detection accuracy of proposed the DPMM approach and the VB, MS, GMM, and K-means approaches.

in detecting Slowloris attacks. This is due to the fact that Slowloris is a stealthy attack, and it is considerably challenging for anomaly detectors to precisely detect such attacks.

Fig. 8 illustrates the normal and aggregated attack traffic detection accuracies. It is evident from Fig. 8 that the proposed DPMM approach outperforms all other approaches in accurately detecting normal traffic applications. However, VB and other approaches struggle to detect HTTP traffic. Note that in Figs. 6 to 8, the results relate to the identification the cluster set U_{fp} of unknown traffic applications from Algorithm 3. We observed that Algorithm 4, which is based on DPMM, performs significantly better than the other approaches in detecting normal and attack traffic applications, with an overall accuracy of over 97%.

E. Flow-level feature-based normal traffic identification

Here, we demonstrate the normal application identification accuracy for the cluster set U_{fp} of unknown fingerprints. The proposed DPMM-based clustering method outperforms all other approaches in classifying various "unknown" fingerprinted applications, as shown in Table VII. The overall classification accuracy from Algorithm 4 for various applications is 96.5%, whereas all other approaches struggle to identify some applications. For instance, GMM performs highly effectively on many applications but is unable to identify WoW game traffic. The reason for this is the dynamic behavior of game application over time. However, the proposed approach identifies all applications with an accuracy of between 71% and 100%. It is also observed that all approaches, i.e., DPMM, VB, MS, GMM, and K-means, classify the CBR traffic with 100% accuracy. The reason for this lies in the fact that the CBR traffic is finely separated in the three-dimensional feature space, as shown in Fig. 5, and after fingerprinting the KL divergence is higher between CBR and other applications.

F. Packet- and flow-level features based fingerprinting

Here, we investigate the performance of our approach based on the fingerprints generated from the packet- and flow-level traffic features. In this set of experiments, we merge packet- and flow-level features to generate fingerprints for different applications, and then analyze the performance using DPMM clustering approach. We are interested to observe the performance of our approach when application fingerprints are generated from both the packet- and flow-level features rather than fingerprints generated using only packet-level features.

We used total eight traffic applications, six were normal traffic and two of them were attack traffic. The normal traffic applications include HTTP, UDP, VoIP, Game, P2P, and TCP, whereas attack traffic applications are DDoS attacks. From the experimental results, we observe that DPMM achieves 66.6% accuracy in detecting normal traffic applications, whereas the detection accuracy for DDoS attacks are 70% and 92%, respectively. The overall accuracy of the proposed approach in detecting normal and attack traffic lies at 70.25%. This proves our claim in Section III-B that by merging packet- and flow-level features and applying classification algorithm on the merged feature may result in misclassifications, i.e., traffic from different applications may result in a same cluster.

VII. CONCLUSION

In this paper, we proposed a novel framework for generating normal traffic profiles based on application fingerprinting. Moreover, we proposed a Bayesian nonparametric traffic application clustering approach to address the problem of unknown application detection. The proposed method introduced two novel approaches. First, to fingerprint traffic application behavior based on packet-level traffic features and derive a multi-modal probability distribution to generate normal traffic profiles. Second, based on the Bayesian nonparametric clustering approach, the unknown traffic applications are identified either as DDoS attack traffic or normal traffic. We had shown that the proposed method classifies unknown normal and DDoS attack traffic applications with accuracies of 100% and 97%, respectively. In this work, we analyzed and successfully detected HTTP, FTP, and DNS applications for the case of normal traffic applications, and DDoS (flooding using IRC botnet or Slowloris) attack applications for the attack traffic scenario.

In future work, we plan to explore and include flash events in our proposed framework. It should be noted that flash events are quite similar in behavior to DDoS attacks. It would be worthwhile studying the behavior of flash events and distinguishing them from DDoS attacks while utilizing the proposed framework.

REFERENCES

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

[2] S. Hiltom, "Dyn Analysis Summary Of Friday October 21 Attack," *[Online]https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/* Gartner, Oct. 2016.

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

[4] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli, "Traffic Classification through Simple Statistical Fingerprinting," in *ACM SIGCOMM Computer Communication Review,* vol. 37, no. 1, pp. 7-16, Jan. 2007.

[5] M. H. Bhuyan, D. K. Bhattacharyya, J. K. Kalita, "A multi-step outlier-based anomaly detection approach to network-wide traffic," in *Elsevier Information Sciences,* vol. 348, no. 2016, pp. 243-271, Feb. 2016.

[6] B. Anderson and D. McGrew, "Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity," in *The 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* pp. 1723-1732, 2017.

[7] A. H. Hamamoto, L. F. Carvalho, L. D. H. Sampaio, T. Abrão, and M. L. Proença Jr, "Network Anomaly Detection System using Genetic Algorithm and Fuzzy Logic," in *Expert Systems with Applications,* vol. 92, pp. 390-402, 2018.

[8] G. Fernandes Jr, L. F. Carvalho, J. Rodrigues, and M. L. Proença Jr, "Network anomaly detection using IP flows with principal component analysis and ant colony optimization," in *Journal of Network and Computer Applications,* vol. 64, pp. 1-11, 2016.

[9] V. Matta, M. M. Di, and M. Longo, "DDoS attacks with randomized traffic innovation: botnet identification challenges and strategies," in *IEEE Transactions on Information Forensics and Security,* vol. 12, no. 8, pp. 1844-1859, 2017.

[10] Z. Tan, A. Jamdagni, X. He, P. Nanda, and R. P. Liu, "A system for denial-of-service attack detection based on multivariate correlation analysis," in *IEEE transactions on parallel and distributed systems,* vol. 25, no. 2, pp. 447-456, 2014.

[11] U. Ben-Porat, A. Bremler-Barr, and H. Levy, "Vulnerability of network mechanisms to sophisticated DDoS attacks," in *IEEE transactions on computers,* vol. 62, no. 5, pp. 1031-1043, 2013.

[12] J. François, I. Aib, and R. Boutaba, "FireCol: a collaborative protection network for the detection of flooding DDoS attacks," in *IEEE/ACM Transactions on Networking (TON),* vol. 20, no. 6, pp. 1828-1841, 2012.

[13] T. P. Fries, "Classification of Network Traffic Using Fuzzy Clustering for Network Security," in *Industrial Conference on Data Mining,* pp. 278-285, 2017.

[14] B. Stone-Gross, et al., "Your Botnet Is My Botnet: Analysis of a Botnet Takeover," in *ACM Conf. Computer Comm. Security,* 2009.

[15] C. Y. Cho, J. Caballero, C. Grier, V. Paxson, and D. Song, "Insights from the Inside: A View of Botnet Management from Infiltration," in *Third USENIX Conf. Large-Scale Exploits and Emergent Threats: Botnets, Spyware, Worms, and More (USENIX LEET),* 2010.

[16] V. L. L. Thing, M. Sloman, and N. Dulay, "A Survey of Bots Used for Distributed Denial of Service Attacks," in *Proc. SEC,* pp. 229-240, 2007.

[17] J. Jung, B. Krishnamurthy, M. Rabinovich, "Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites," in *The 11th international conference on World Wide Web,* pp. 293-304, May 2002.

[18] M. E. Ahmed, H. Kim, and M. Park, "Mitigating DNS Query-Based DDoS Attacks with Machine Learning on Software Defined Networking," in *The 36th IEEE Military Communications Conference,* Oct. 2017.

[19] Y. Gu, A. McCallum, D. Towsley, "Detecting Anomalies in Network Traffic Using Maximum Entropy Estimation," in *ACM SIGCOMM conference on Internet measurement,* Oct. 2005.

[20] A. Scherrer, N. Larrieu, P. Owezarski, P. Borgnat, and P. Abry, "Non-Gaussian and Long Memory Statistical Characterizations for Internet Traffic with Anomalies," in *IEEE Trans. Dependable and Secure Computing,* vol. 4, no. 1, pp. 56-70, Jan. 2007.

[21] S. Behal and K. Kumar, "Detection of DDoS attacks and flash events using information theory metrics," in *Elsevier Computer Networks,* vol. 116, pp. 96-110, Apr. 2017.

[22] S. Yu, W. Zhou, W. Jia, S. Guo, Y. Xiang, and F. Tang, "Discriminating DDoS Attacks from Flash Crowds Using Flow Correlation Coefficient," in *IEEE Transactions on Parallel and Distributed Systems,* vol. 23, no. 6, pp. 1073 - 1080, Jun. 2012.

[23] M. E. Ahmed, D. I. Kim, J. Y. Kim, and Y. A. Shin, "Energy arrival-aware detection threshold in wireless-powered cognitive radio networks," in *IEEE Trans. Vehic. Technol.*, vol. 66, no. 10, pp. 9201 - 9213, 2017.

[24] D. J. C. MacKay, "Information Theory, Inference and Learning Algorithms," in *Cambridge University Press*, 2003.

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

[26] F. Wood and M. J. Black, "A nonparametric Bayesian alternative to spike sorting," in *Journal of Neuroscience Methods*, vol. 173, no. 1, pp. 1-12, Jun. 2008.

[27] B. S. Everitt, D. J. Hand, "Finite mixture distributions," in *Chapman & Hall*, ISBN 0-412-22420-8, pp. 790-799, 1981.

[28] J. Zhang, C. Chen, Y. Xiang, W. Zhou, and A. V. Vasilakos, "An Effective Network Traffic Classification Method with Unknown Flow Detection," in *IEEE Transactions on Network and Service Management,* vol. 10, no. 2, pp. 133-147, Jun. 2013.

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

[30] The CAIDA UCSD DDoS attack 2007, "Online: http://www.caida.org/data/passive/ddos20070804_datasetẋml," 2007.

[31] CAIDA dataset, "Online: https://www.caida.org/data/," 2016.

[32] [Online]. Available: "https://crawdad.org/kaist/wibro/20080604/,".

[33] [Online]. Available: "https://crawdad.org/snu/wow_via_wimax/20091019/,".

[34] [Online]. Available: "https://crawdad.org/snu/bittorrent/20110125/,".

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

[36] K. P. Murphy, "Conjugate Bayesian analysis of the Gaussian distribution," (PDF) 2007.