Copyright © 2018 Tech Science Press CMC, vol.55, no.1, pp.95-119, 2018

An Abnormal Network Flow Feature Sequence Prediction Approach for DDoS Attacks Detection in Big Data Environment

Jieren Cheng[1, 2], Ruomeng Xu[1, *], Xiangyan Tang[1], Victor S. Sheng[3] and Canting Cai[1]

Abstract: Distributed denial-of-service (DDoS) is a rapidly growing problem with the fast development of the Internet. There are multitude DDoS detection approaches, however, three major problems about DDoS attack detection appear in the big data environment. Firstly, to shorten the respond time of the DDoS attack detector; secondly, to reduce the required compute resources; lastly, to achieve a high detection rate with low false alarm rate. In the paper, we propose an abnormal network flow feature sequence prediction approach which could fit to be used as a DDoS attack detector in the big data environment and solve aforementioned problems. We define a network flow abnormal index as PDRA with the percentage of old IP addresses, the increment of the new IP addresses, the ratio of new IP addresses to the old IP addresses and average accessing rate of each new IP address. We design an IP address database using sequential storage model which has a constant time complexity. The autoregressive integrated moving average (ARIMA) trending prediction module will be started if and only if the number of continuous PDRA sequence value, which all exceed an PDRA abnormal threshold (PAT), reaches a certain preset threshold. And then calculate the probability that is the percentage of forecasting PDRA sequence value which exceed the PAT. Finally we identify the DDoS attack based on the abnormal probability of the forecasting PDRA sequence. Both theorem and experiment show that the method we proposed can effectively reduce the compute resources consumption, identify DDoS attack at its initial stage with higher detection rate and lower false alarm rate.

Keywords: DDoS attack, time series prediction, ARIMA, big data.

1 Introduction

In the recent years, with the development of big data, there are various data on the network. According to Cisco visual networking index in 2016 [Cisco, VNI (2016)], global IP traffic was 1.2 ZB per year or 96 EB (one billion Gigabytes [GB]) per month. By 2021, global IP traffic will reach 3.3 ZB per year, or 278 EB per month. Therefore, it is crucial that the detection method distinguishes accurately and timely between normal network flow and cyber-attack with limited compute resources. Currently, Distributed

[1] School of Information Science and Technology, Hainan University, 570228, Haikou, China.

[2] State Key Laboratory of Marine Resource Utilization in South China Sea, 570228, Haikou, China.

[3] Department of Computer Science, University of Central Arkansas, Conway, AR 72035, USA.

* Corresponding author: Ruomeng Xu. Email: 0xbbc@0xbbc.com.

Denial of Service (DDoS) attack has been one of the greatest threats to network security for years. DDoS attacks are attackers who use client or server technology to unite multiple computers as an attack platform to attack one or more targets to enhance the effect of the attack. DDoS attacks cause the victim host or network cannot timely receive and process external normal requests, or cannot respond to external normal requests in a timely manner, thus failing to provide normal services to legitimate users. DDoS attacks have a wide range of damage, increasing the complexity of attacks. To reduce the harm of DDoS attacks to the network, a great deal of DDoS attack detection methods have been put forward both in academia and industry. However, with the continuous development of network technology, DDoS attacks are becoming more and more complex and diverse, which makes it difficult for some DDoS attacks based on single network traffic to detect multiple types of DDoS attacks. Currently, DDoS attack detection method has the problems of single type of detection attack, missing report and false alarm. Moreover, DDoS attack detection methods cannot identify the network status in time because of the complexity of the algorithms. In order to resist DDoS attacks more effectively, DDoS defense mechanisms should be able to detect and predict quickly while allowing legitimate users to access the target, especially in the big data environment. Currently, DDoS attacks are so rampant that attacks can occur anywhere in the world. At present, more and more studies are devoted to DDoS attacks, it becomes increasingly difficult to detect fast and accurately that whether there is a DDoS attack or not due to the diversification of DDoS attacks.

In this paper, we firstly extract, transform and load the normal network flow from the big data we collected into the workspace. Then the proposed method learns from the normal network flow and build a model that measures the network flow which needs detection, the result is represented by the network flow abnormal index value. We collect network packets from the detection network flow with a certain time interval, and calculate the feature value at the end of each time interval. The feature value is basically based on the number of new users and old users, average accessing rate of each new user, and the number of old users in training sample sets respectively in every time sampling interval. This feature value we proposed is named $PDRA$, or to be specific, network flow abnormal index value. And it is designed to measure the status of current network flow.

To effectively decrease the consumption of compute resources and minimize the interference with normal network flow, we only start the ARIMA module with sliding window mechanism for trending prediction if there are abnormal sample points been detected. To maximize the time for the prepare of DDoS attack defense and give a timely respond on abnormal network flow, we calculate the possibility of current network flow is DDoS attack, confirm it is a DDoS attack or plainly some fluctuation of the network flow. Through experiments, we verify the correctness of this method, and it has high accuracy on distinguishing normal flow, DDoS attack flow and hotspot events with low false alarm rate.

During these processes, as a matter of fact, the amount of the data required for training, analyzing, feature value computing and trending prediction is all relatively small, and the computation complexity of our method is low. This allows our method to work effectively when using in the big data environment. And we could distribute the sampling and calculation task to multiple nodes in the cluster, then merge the result for trending prediction.

The rest of the paper is organized as follows. Related work is discussed in Section 2. In the Section 3, we will detail the calculation method of the network flow abnormal index value $PDRA$ and characterized each part of it. In the analysis, we will explain how the network flow abnormal index value could indicates different types of network flow, i.e. normal network flow, DDoS network flow and hotspot events, based on the characteristic value of $PDRA$. We will then introduce time-series model for trending prediction with the application of sliding window mechanism in our DDoS attack detector, then we describe how does the ARIMA module would be active for trending prediction, and how does this activation mechanism save compute resources without latencies of detection in big data environment. In Section 4, we use the CAIDA DDoS Attack 2007 as our dataset, and we evaluate the proposed method with different parameters to verify that this method is effective and could effectively distinguish normal network flow, DDoS attack flow and hotspot events. And lastly comes the Section 5, we conclude the advantage of the method we proposed, and discuss some further work for the enhancement of this method.

2 Related works

Presented in STONE [Vincenzo, Mar, Zhang et al. (2015)], a framework with expert system functionality that provided effective and joint DDoS detection and mitigation. Moreover, during the mitigation of DDoS attack, it minimizes the degradation of legitimate traffic, which is important in the big data environment. The authors Poongodi et al. [Poongodi and Sundan (2015)] proposed the Aumann agreement theorem based trust re-evaluation in order to reduce the false positive and negative probabilities, such that the accuracy of the system is enhanced. The authors Gurusamy et al. [Gurusamy and Subramaniam (2017)] proposed a machine learning approach for classification, which could be further adapt to our proposed method. The authors Gu et al. [Gu, Sun and Sheng (2017)] proposed the structural minimax probability machine (SMPM) for prediction. The authors Poongodi et al. [Poongodi and Sundan (2015)] proposed approach involves trust-based evaluation wherein the intrusion detection is done using secured trust evaluation policies. A novel IDS is designed using the trust evaluation metrics. This is used for the detection of the flooding DDOS attacks in the networked architecture. Liao et al. [Liao, Li, Kang et al. (2015)] analyzed the differentiation between users' behaviors, extracted two feature sequences from Web logs to represent characteristics of user behavior, and then, application layer DDoS attack detection system architecture based on feature sequences is presented. The authors designed an attack that degrades the performance of an SBS scheduler by subjecting it to a job size distribution which violates the core traffic properties from which SBS derives its strengths [Serwadda and Phoha (2015)]. A scheduling-based architecture is proposed for the SDN controller that leads to effective attack confinement and network protection during denial of service (DoS) attacks [Lim, Yang, Kim et al. (2015)]. Xiao et al. [Xiao, Qu, Qi et al. (2015)] presented an effective detection approach based on K-nearest neighbors traffic classification with correlation analysis (CKNN) to detect DDoS attacks. The authors Kaur et al. [Kaur and Kaur (2017)] proposed a fuzzy approach for the IoT-based automated employee performance appraisal, the framework in their work classifies raw IoT data into three activities, which could also be adapted into our work for the classification of network flow.

The authors Ramanauskaitė et al. [Ramanauskaitė, Goranin, Čenys et al. (2015)] proposed the DDoS attack model, which allows estimation of influence of Botnet size and agent allocation strategies on attack success probability. The authors Seo et al. [Seo and Lee (2016)] calculated the frequency of network-based packet attributes, analyzed the anomalies of the attributes in order to detect IP-spoofed DDoS attacks and proposed a method for the effective detection of malware infection systems triggering IP-spoofed DDoS attacks on an edge network. Malhi et al. [Malhi and Batra (2016)] proposed a secure genetic-based framework for the detection and prevention of masquerade and distributed denial of service attacks in VANETs. The authors Dick et al. [Dick and Scheffer (2016)] applied a known method that sequentially learns the controller and the structured-prediction model. Yan et al. [Yan, Yu, Gong et al. (2016)] discussed the new trends and characteristics of DDoS attacks in cloud computing, and provided a comprehensive survey of defense mechanisms against DDoS attacks using SDN. The authors in Yan et al. [Yan, Gong and Deng (2016)] introduced a solution based on fuzzy synthetic evaluation decision-making model that is effective and lightweight in terms of the resources that it uses. In order to assure the all times availability of patients data, Latif et al. [Latif, Abbas and Latif (2016)] proposed a distributed victim based DDoS attack detection mechanism based on very fast decision tree (VFDT) learning model in cloud-assisted WBAN. The authors Saied et al. [Saied, Overill and Radzik (2015)] have chosen an Artificial Neural Network (ANN) algorithm to detect DDoS attacks based on specific characteristic features (patterns) that separate DDoS attack traffic from genuine traffic. Luo et al. [Luo, Chen, Li et al. (2017)] presented the design, implementation, and evaluation of dynamic PID (D-PID), a framework that uses PIDs negotiated between the neighboring domains as inter-domain routing objects. The authors in Han et al. [Han, Bi, Liu et al. (2017)] proposed a novel DDoS attack detection system based on Spark framework. The authors Yan et al. [Yan, Gong and Yu (2017)] proposed a multi-queue SDN controller scheduling algorithm based on time slice allocation strategy.

Boro et al. [Boro and Bhattacharyya (2017)] presented a defense solution referred to as DyProSD that combines both the merits of feature-based and statistical approach to handle HDDoS flooding attacks. The aim of this systematic survey is to gain insights into the current research on the detection of these attacks by comprehensively analyzing the selected primary studies to answer a predefined set of research questions [Singh, Singh and Kumar (2017)]. It analyzed the various existing detection and prevention approaches that are proposed to tackle ICMPv6-based DoS and DDoS attacks [Elejla, Anbar and Belaton (2017)]. The authors Bawany et al. [Bawany, Shamsi and Salah (2017)] proposed a novel framework for detection and mitigation of DDoS attacks in a large-scale network which comprises a smart city built on SDN infrastructure. Eid et al. [Eid and Aida (2017)] represented a single-vector and a multi-vector complex HTTP(S)-DDoS attack scenarios. The authors Behal et al. [Behal and Kumar (2017)] initiatively proposed using a novel set of information theory metrics called phi-Entropy and phi-Divergence metrics for detecting DDoS attacks and flash events. Park et al. [Park, Yoo, Ryu et al. (2015)]

proposed the DDoS attack model, which allows estimation of influence of B] proposed a DDoS attack path-based collaborative defense mechanism (APCDM) that, by utilizing the management features of a network management system (NMS) or element management system (EMS), detected a DDoS attack path. The authors Subramanian et al. [Subramanian, Gunasekaran and Selvaraj (2015)] proposed work filters majority of the spoofed traffic by Hop Count Filter-Support Vector Machine (HCF-SVM) algorithm on the network layer. Rashidi et al. [Rashidi, Fung and Bertino (2017)] proposed a DDoS defense mechanism named CoFence, which facilitates a “domain-helps-domain” collaboration network among NFV-based domain networks. A real-time DDoS detection method is proposed that uses a novel correlation measure to identify DDoS attacks [Hoque, Kashyap and Bhattacharyya (2017)]. The authors Kim et al. [Kim and Lee (2016)] proposed an information security method to detect and Protect DDoS attacks from the terminal phase using a Preemptive military strategy concept. Jia et al. [Jia, Ma, Huang et al. (2016)] focused on how to distinguish the attack traffic from normal data flows in Big Data and propose a novel real-time DDoS attack detection mechanism based on Multivariate Dimensionality Reduction Analysis (MDRA). The authors Peng et al. [Peng, Leckie and Ramamohanarao (2004)] proposed a simple but robust scheme to detect denial of service attacks (including distributed denial of service attacks) by monitoring the increase of new IP addresses. Peng et al. [Peng and Leckie (2003)] introduced a practical scheme to defend against Distributed Denial of Service (DDoS) attacks based on IP source address filtering.

3 The proposed method

3.1 Basic workflow

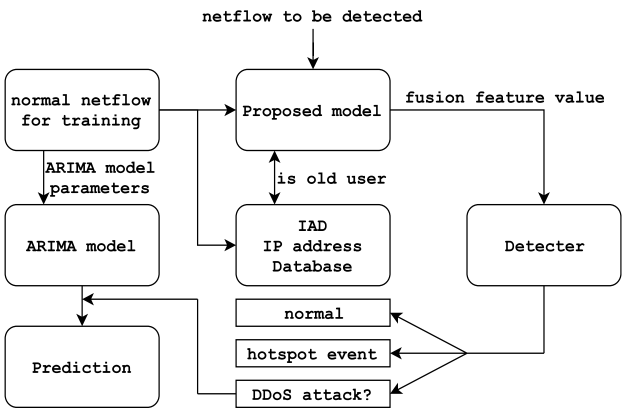

Figure 1: Basic workflow of the proposed method

The basic workflow in the proposed method below is rather simple, we take normal network flow for training, and this procedure outputs our IP address database (IAD) which stores old users, parameters for the proposed method and ARIMA model. Then the network flow to be detected can pour into our method, we'll inquire the IAD for checking whether an IP address is our old user or not. And we calculate the PDRA and pass it to our classifier. The detector could identify three types of network flow, namely, normal, hotspot event and DDoS attack. If the output of the classifier is DDoS attack, then we activate our prediction procedure using ARIMA with pre-trained parameters. The whole workflow is shown in Fig. 1.

3.2 Problem formulation

Given a normal network flow U with n sample IP packets and a network flow V with m IP packets which need to be detected, we define each IP packet as $(\mathrm{T_i}, \mathrm{S_i}, \mathrm{D_i})$, where T_i is the arrival time of the packet i, $S_i$and D_i denotes its source IP and the destination IP respectively. We will use the normal network flow U to generate some parameters which uses for identification and prediction for V, and we will take a same Δt as a parameter for both training and detection algorithm.

With above definition, in the training procedure, we collect a subgroup of sample IP packets G_k from normal network flow U within the k-th Δt. Once we have reached the end of every Δt, a filter is then applied to G_k to drop out all packets which has an invalid public IPv4 address of its S. Thus the filtered sample group F_k is defined as.

$$\forall_{G_{k_j}} \in G_k$$
$$G_{k_j} \in F_K \text{ if } S_{G_{k_j}} \text{ is valid public IPv4 address} \tag{1}$$

While obtaining each F_k, we incrementally build an IP address set O which denotes our old users. We union all S in F_1 into O in the first time interval, $O_{max} = \{\} \cup \{S|S \in F_1\}$, and let $O_{max} = || \{S|S \in F_1\} ||$. Then for each successive F_k, it is obviously that there are $F_k \cap O$ many old users in the k -th time interval, and we update $O_{max} = \max(O_{max}, ||F_k \cap O||)$.

$$\forall_{F_k}$$
$$O_{max} = \begin{cases} ||F_1||, & k = 1 \\ \max(O_{max}, ||F_k \cap O||), & k \geq 2 \end{cases} \tag{2}$$

After we calculated the O_{max}, we merge the F_k into O. By continuously doing these steps, finally we will get an O_{max} which represents the maximum number of old user appeared in some certain Δt.

Meanwhile the updating of O_{max}, we compute an $N_k = ||F_k|| - ||F_k \cap O||$. Because we've defined that set $F_k \cap O$ denotes our old user, then correspondingly, the set $F_k \setminus O$ would denote new users, and N_k is the number of new user. By the time we get the final O_{max}, we calculate our average number of new user over all the time intervals as

$$\overline{N} = \frac{\sum_{i=1}^{k} N_i}{k} \tag{3}$$

When we have acquired these four basic model parameters, namely, Δt, O, O_{max} and $\overline{N}$, we could process the detection procedure. We read the network flow V using the same Δt, and during each time interval k, we store all the source IP S in the dictionary W_k with S as the key and the times it occurs as the value, $W_k = [S_{k,i}, o_{k,i}]$. Then we let set G_k equals to all the keys in W_k, and we define the $W_k[S_{k,i}]$ as the times of the corresponding source IP $S_{k,i}$ occurred in the k-th time interval. At the end of each Δt, we calculate four features of the k-th time interval. We define four features below.

Definition 1. The percentage of old user appeared in current k-th Δt over the maximum of our old user of some certain time intervals. We define this percentage as P_k

$$P_k = \frac{\left||G_k \cap O|\right| - O_{max}}{O_{max}} \tag{4}$$

Definition 2. The amount of changes of our new users in amount compared with the amount of the average new user $\bar{N}$. We define this amount as D_k.

$$D_k = \left||G_k \setminus O|\right| - \bar{N} \tag{5}$$

Definition 3. The ratio of current new users to our maximum old users.

$$R_k = \frac{||G_k \setminus O||}{O_{max}} \tag{6}$$

Definition 4. Current access rate of new users. To be specific, it denotes that there are average A_k accessing requests per second per new user. We define this average accessing rate per new user per second as A_k.

$$A_k = \frac{\sum\{W_k[S_{k,j}]|\forall_{S_{k,j}} \in (G_k \setminus O)\}}{||G_k \setminus O||\Delta t} \tag{7}$$

Definition 5. Within P_k, D_k, R_k and A_k defined, we firstly calculate the product of them, then we take the negative value of the product. This negative value is defined as $PDRA_k$.

$$PDRA_k = -P_k D_k R_k A_k \tag{8}$$

Assumption on $\boldsymbol{P}$. With the observation of real-life in the big data era, during normal time, our old users appear in a relatively fixed pattern. And this observation suggests that we could take this prior as a measurement during the DDoS detection.

For instance, suppose we observed that $O_{max} = 2 \times 10^6$ and given a certain Δt in detection network flow, if there're $\left||G_k \cap O|\right| = 1 \times 10^6$ old users, then P is calculated as $P = \frac{1\times10^6 - 2\times10^6}{2\times10^6} = -0.5 = -50\%$. Because P is a signed float number, it can indicate either the increment or decrement of number of current old users compared with the amount of maximum old users in percentage.

Assumption on $\boldsymbol{D}$. Based on the definition of DDoS, we assume that if we are under a DDoS attack, there should be a great many of new users, therefore $||G_k \setminus O||$ is supposed to be much larger than the amount of average new users $\bar{N}$ in normal time. And this factor D is used as a quantity which represents the delta between the number of observed new users and the amount of average new users.

Assumption on $\boldsymbol{R}$. If a DDoS attack is in progress, it is reasonable to presume that the amount of current new users $||G_k \setminus O||$ is larger than the amount of current old users $||G_k \cap O||$. However, time zone exists and new users may come from all over the world. Chances are that there're just a few old users and for whatever reason a small group of new users appears, in such case, if the amount of new users is larger than the old ones, using $\frac{||G_k \setminus O||}{||O||}$ will cause fluctuation. Thus we have using the ratio of the amount of current new users to observed amount of maximum old users in training.

Assumption on *A*. Based on assumptions D and R, if there is a DDoS attack, then the average accessing rate from new users should be a large number since they are sending flood packets. Nevertheless, if it is a normal network flow or hotspot event, because every normal user should obey the TCP/IP principles, thus the average accessing rate should be a relatively small constant value.

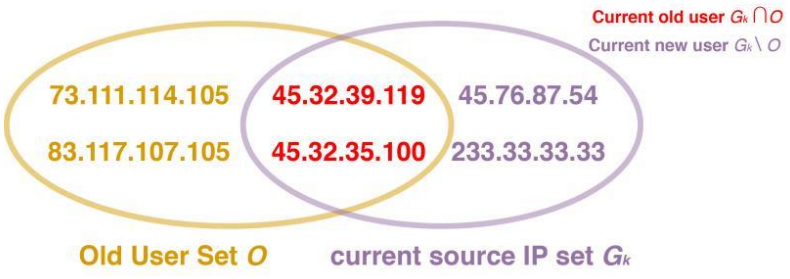

Figure 2: Set of current old users and set of current new users

Noting that we do not filter out invalid IP address in G_k, as Fig. 2 shows, because in the real-life DDoS attack network flow, attackers may fabricate IP address which is an invalid public IP. If we drop these invalid IP, we're actually shrinking the attack. However, when we calculate the old user set O from training network flow U, we do need to filter out all invalid IP addresses to ensure the correctness, which is, old users shall come from a valid public IP address.

The training algorithm is shown in Algo. 1.

Based on our assumptions P, D, R and A, we can identify three kinds of network flow, namely,

1) Normal. In this case, we expect the number of current old users $||G_k \cap O||$ to be around the amount of maximum old users O_{max}, and thus leading the P to approach ± 0. Then the number of current new users should be also around the average amount of new users $\bar{N}$, and this gives us a D which also approaches 0. As for R, empirically, old users are majority compared with new users, and this suggests that R would be a

Algorithm 1 Training

```
Input: normal network flow $N$
Output: PDRA value, $O_{max}, \bar{N}$
 1: $t \leftarrow 0$
 2: while has next data do
 3:     get next data $(T_i, S_i, D_i)$ from dataset
 4:     move to next data if $S_i$ is invalid
 5:     $F \leftarrow F \cup \{S_i\}$
 6:     update $t$
 7:     if $t == \Delta t$ then
 8:         update $O_{max}$ according to Eq. 2
 9:         update $\bar{N}$ according to Eq. 3
10:         calculate PDRA according to Eq. 8
11:         $t \leftarrow 0$
12:     end if
13: end while
14: return PDRA, $O_{max}, \bar{N}$
```

number which approaches to $+0$. Lastly, accessing rate A_k per second per new user should be a small constant value c. Then we multiply them together, the final result should be very close to ± 0.

$$\begin{aligned} &\because P_k \to \pm 0, D_k \to 0, R_k \to +0, A_k = c \\ &\therefore PDRA_k = -P_k D_k R_k A_k \to \pm 0 \end{aligned} \tag{9}$$

For website oriented to local users, there would be one more sub-case, inactive time. We use the term inactive time to denote periods like night time, rest time, lunch time so on and so forth according to the specific situation. During such periods, the number of old users would be relatively small, and that will lead the value of P to approach to -0.5 or even the worst-case, -1. But periods like these won't be a exception in our model, because D and R are still approaching to 0, the final value would still stick at somewhere around 0.

2) DDoS. If we are under DDoS attack, we expect the number of current new users $||G_k \setminus O||$ is much larger than its average $\bar{N}$, and this results in a large value of D. For a valid DDoS attack, the website or the network would be nearly unavailable to serve old users. With this real-life experience, the amount of current old users $||G_k \setminus O||$ would be a very small number, and this leads the value of P to be close to -1. And accessing rate A_k from new users is a large number. Now, in this case, we have

$$\begin{aligned} &\because P_k \to -1, D_k \gg 1, R_k > 0, A_k \gg 1 \\ &\therefore PDRA_k = -P_k D_k R_k A_k \gg +1 \end{aligned} \tag{10}$$

Furthermore, P could be viewed as the DDoS impact on old user, and the product $PDRA_k$ could be used as an overall DDoS attack impact indicator. The reason for the former one is obvious, and for the latter, we could interpret our Eq. (10) as the overall DDoS impact value calculated as the number of D new users multiply by the percentage of old users P we cannot serve, and then multiply by current DDoS scale factor R.

3) Congestion. Yet another network flow we are able to detect is network congestion. When a hot topic appears, we can observe that both the number of new users and old users largely increased. There are three implications obtained from this observation, firstly, here we got a lot more new users in this case, thus D should be a large positive value. Second, because it is a hot topic, it is reasonably to presume that old users would also focus on it, therefore the value of P would be larger than 1. And finally, even if there're a great many of new users, they normal user always obey the TCP/IP principle, thus it would result in a relatively small constant value c.

$$\begin{aligned} &\because P_k > 1, D_k \gg 1, R_k > 0, A_k = c \\ &\therefore PDRA_k = -P_k D_k R_k A_k \ll -1 \end{aligned} \tag{11}$$

Moreover, R_k is not only a scale factor in this case, but also describes in what degree do the external users concern about this hot topic. If $R_k \in (0, 1]$, it may suggests that old users are more concerning about current event. Otherwise, it implies that external new users are more interested in this topic.

3.3 Refinement

Since we are using a product of three aforementioned features, it is important to avoid 0 as factor in our Eq. (10). Thus small changes have been made to the calculations of P_k and R_k.

$$P_k = \begin{cases} \dfrac{||G_k \cap O|| - O_{max}}{O_{max}}, & \text{if } ||G_k \cap O|| - O_{max} \neq 0 \\ \dfrac{||G_k \cap O|| - O_{max} - 1}{O_{max}}, & \text{otherwise} \end{cases} \tag{12}$$

$$R_k = \begin{cases} \dfrac{||G_k \setminus O||}{O_{max}}, & \text{if } ||G_k \setminus O|| \neq 0 \\ \dfrac{||G_k \setminus O|| - 1}{O_{max}}, & \text{otherwise} \end{cases} \tag{13}$$

And to deal with the huge amount of the accessing and comparison of IP addresses, we designed a fast IP address database which could flag, unflag and check for an IP address in $O(1)$ time. The original string format of IP comparison would take $O(nm)$ time, because there are $O(n)$ IP in our training set, and $O(m)$ IP in our detection set. And in big data environment, both n and m could be a large number, which makes it a pseudo polynomial time algorithm, especially when IPv6 is widely used. Even if we store the IPv4 address in raw format, i.e. unsigned 32 bit integer that will take literally 4 GB space, let alone the storage of IPv6. However, we could use consecutive bits for our flags, which mean that it only requires 512 MB space for the whole IPv4 addresses. The application of this IP address database could significantly reduce the time which spent on IP address matching, we use a reasonable size of memory space to ensure the lower time complexity, i.e. constant access time for reading, writing and checking whether an IP address has been flagged in the database or not.

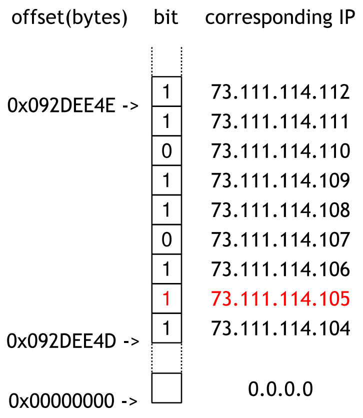

Figure 3: IP address database example

For instance, if we have the IAD set as Fig. 3, and we currently want to check that whether the IP address 73.111.114.105 had been flagged, we could simply compute the corresponding offset in byte and bit by Eq. (14). In raw data, the IP address 73.111.114.105 is presented as an unsigned int, which is 1232040553. To get the offset in byte, we divide 1232040553 by 8, and the result is 154005069 in decimal, i.e. 0x092DEE4D in hexadecimal. And the offset in bit is the modular we just computed, which is 1. So we should check the second bit at 0x092DEE4D, if it was set to 1, then this IP is flagged as old IP, otherwise it is a new IP.

$R_{\mathrm{IP}} \leftarrow$ raw IPv4 representation in unsigned int

$$\text{offset}\begin{cases}\text{byte}, & R_{IP} \quad | \, 8\\ \text{bit}, & R_{IP} \; mod \; 8\end{cases} \tag{14}$$

As we can see, there is only one dividing and one modular operation, and we could further optimize this part with bitwise manipulations. Because R_{IP} is divided by 8, we could just right shift 3 bits, and if we would like to compute the modular of $R_{IP} \; mod \; 8$, we could use a bit mask for it. This would make it perform better.

$R_{\mathrm{IP}} \leftarrow$ raw IPv4 representation in unsigned int

$$\text{offset}\begin{cases}\text{byte}, & R_{IP} \quad \gg \; 3\\ \text{bit}, & R_{IP} \;\; and \; 0\text{x}7\end{cases} \tag{15}$$

At the same time, the IAD is not only scalable for IPv6 if we use good hash algorithm to map the 128 bit IPv6 address to a 32 bit integer, but also practically works. Because even we set 25% as our capacity bar of the hash table, it could hold 2^{30} different addresses.

3.4 DDoS detection with time series

3.4.1 Abnormal network flow time-series modeling

As the CAIDA DDoS attack 2007 dataset describes, the duration of each network flow trace in the dataset is 5 min, the beginning of the whole network flow trace starts at 13:49:36, DDoS attack occurs at around 14:15:56. So we took the trace which the beginning of the DDoS attack located in as our detection network flow here. As the CAIDA DDoS attack 2007 dataset describes, the duration of each network flow trace in the dataset is 5 min, the beginning of the whole network flow trace starts at 13:49:36, DDoS attack occurs at around 14:15:56. So we took the trace which the beginning of the DDoS attack located in as our detection network flow here.

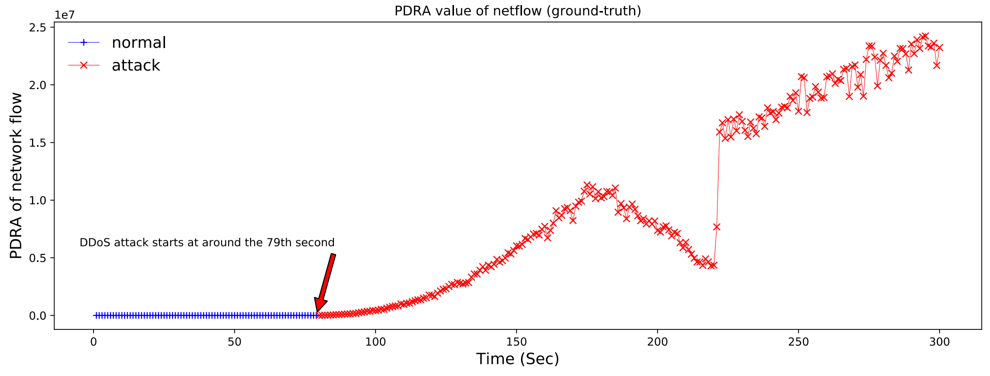

Figure 4: Ground truth of CAIDA DDoS attack dataset 2007

The training set should be as pure as possible, we could apply more data clean techniques while doing the ETL for the training set, and there are many sophisticated data clean techniques in the big data field, to get a better training set is one out of many factors which improves the accuracy of the following algorithm, but we will not detail these data clean techniques since they beyond this paper's scope. In our work, we moved the data clean task into the training procedure, we dropped all IP packets which without a valid IPv4 address. With a good training set and the proposed $PDRA$, we further take the ARIMA model into our work for trending prediction.

We sampled and calculated the $PDRA$ value of the normal DDoS attack network flow with Δt, and after N samples, we get a time series $K = PDRA_i, i = 1,2,\cdots,N$.

We build our ARIMA trending prediction module with R language, as we can see that the original data is not stationary, and the Augmented Dickey-Fuller Test (ADF) test confirms that. The possibility P of original is not stationary is 98%. To avoid the tendency, we had taken two times differential of original data, then processed data passed ADF test.

Table 1: Augmented dickey-fuller test

	Dickey-Fuller	Lag order	P
unit	-11.7260	6	$\ll 0.01$
white noise	-6.2332	6	$\ll 0.01$

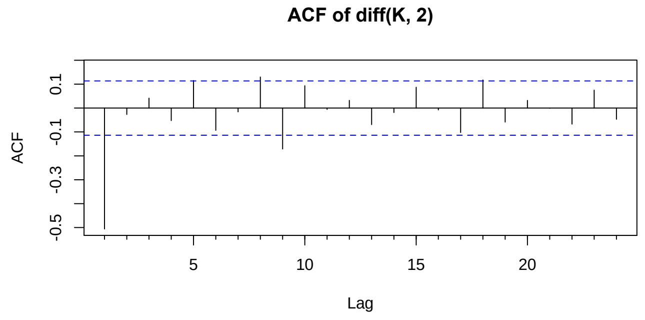

Figure 5: ACF on time series K

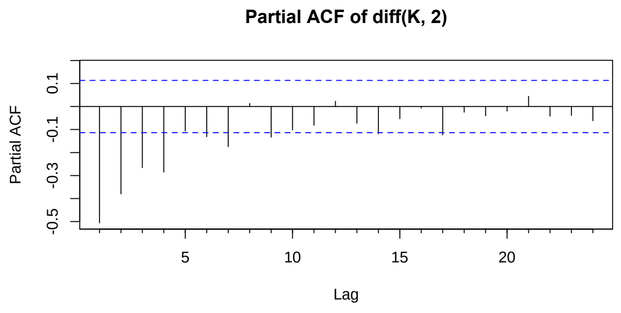

Figure 6: Partial ACF on time series K

As we can see from Tab. 1, because the probability of the unit is less than 0.05, thus makes it a white noise, and our differential order $d = 2$. With this differential order, we look into the autocorrelation and partial correlation as shown Fig. 5 and Fig. 6. As Akaike Information Criterion suggests, the model we build should be ARIMA (1, 2, 2).

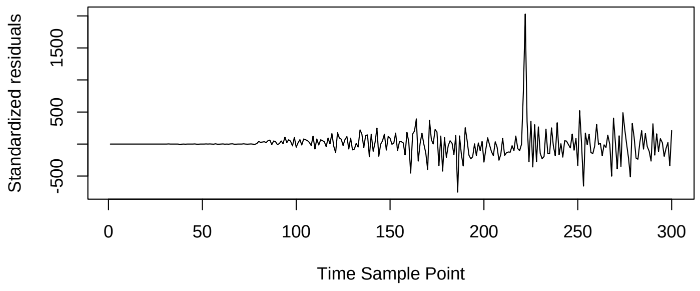

Figure 7: Standardized Residuals

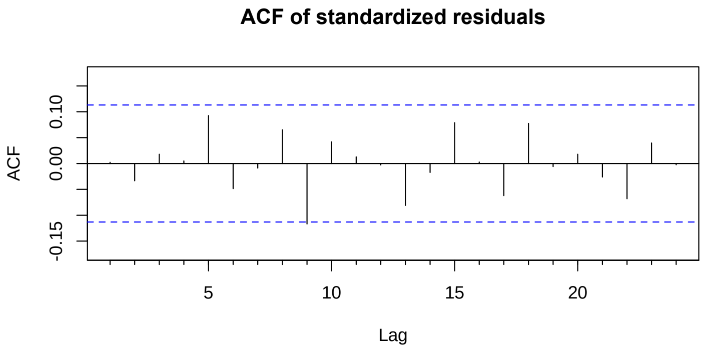

Figure 8: ACF of Standardized Residuals

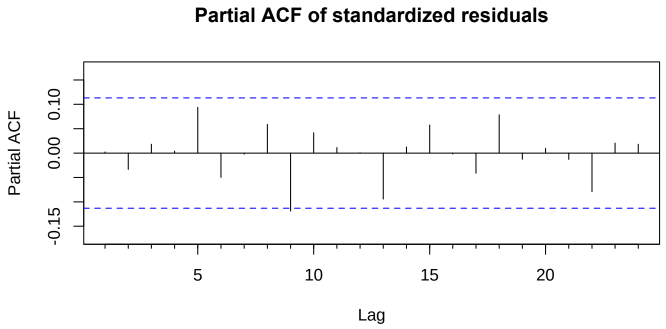

Figure 9: Partial ACF of Standardized Residuals

To further validate our model, we test that whether the residuals is white noise or not by Ljung test. And the time series, ACF and partial ACF of residuals is shown in Fig. 7, Fig. 8 and Fig. 9 respectively.

In our observation Fig. 13, normal network flow (first 80 seconds around) shows a very small value, and when DDoS attack comes, the $PDRA$ increases sharply, and it is always a big number during the DDoS attack, thus we can distinguish DDoS attack from normal network flow. The naive classifier we used is simply based on a threshold value α and requires β consecutive sample points exceed the threshold to trigger the alarm. According to our observation, the maximum $PDRA$ is about 15, and the minimum $PDRA$ is 0, therefore we choose the threshold value $\alpha = \max + (\max - \min) = 15 + (15 - 0) = 30$ from both statistics and tolerant perspectives for all different Δt, and in this paper, we set $\beta = 2$, i.e. there should be 2 consecutive abnormal sample points to trigger our DDoS alarm. Because if we trigger the ARIMA trending prediction module with only 1 abnormal point detected, then it is highly possibly that it is only the fluctuation, and this would cause unnecessary consumption of compute resources. Therefore it is no meaning to select 1 as the value of β. And we could select a larger β for different demanding of sensitivity or different available compute resources, besides, a higher value for β would ease a longer fluctuation of normal network flow. We select 2 as the value of β, because it is the minimum meaningful value for β, and it has the minimum effect of easing the fluctuation of normal network flow, thus it could be used as a baseline.

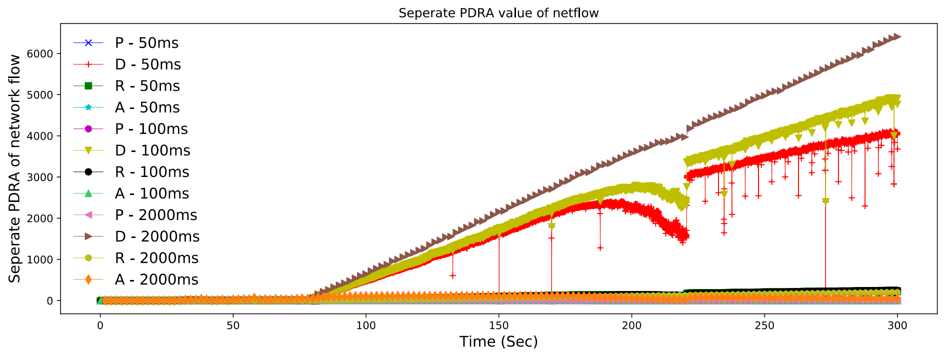

Figure 10: Analysis of the big drop of PDRA

The big drop around 220 second is quite noticing, thus we plot our internal value, i.e. plot P, D, R and A separately, as shown in Fig. 10. Within the experiment result, the big drop mainly comes from the factor D. Because the factor D we defined as the difference between the number of average new user and current new user, it is a numerical value, and in real-life, it can varies in very short time. Another finding observed in this experiment is that, if we take a shorter Δt, the factor D is going to be more sensitive on the difference between the number of average new user and current new user. However, when we consider all four factors, the high sensitivity of factor D resulted in relatively low accuracy of the final prediction.

3.4.2 Abnormal triggered time-series trending forecast

Our proposed method uses ARIMA trending prediction module to improve the accuracy of prediction. Basic idea is that, if we detected abnormal value of $PDRA$, then we would trigger the pre-trained ARIMA for trending prediction. Our trending prediction module with ARIMA mainly uses a sliding window which contains the most recent w points. If the abnormal $PDRA$ value in the prediction given by ARIMA module exceeds a certain percentage, the DDoS attack alarm should be triggered. Otherwise, we will continuously make forecast of current network flow until stop conditions are met.

And big data environment requires the detection procedure to be not only able to identify the DDoS attack fast with high accuracy, but also have a relatively low consumption of compute resources, and avoid or minimized the interference with normal network flow. Given that the normal network flow might as huge as hundreds of gigabytes per second, we designed an activation method for the trending prediction, which is inactive in normal network flow, and it is triggered to be active only when abnormal sample points are detected. And finally, if ARIMA trending prediction module could confidentially identifies that there is a DDoS attack, we will pause the ARIMA trending prediction module. The detailed and formal detection procedure is described as following.

During the detection, we take α as the threshold value, and if there are consecutive β sample points exceeds the threshold. We define these β points as abnormal points, then we will do the trending predictions with pre-trained ARIMA module. Let $N = [N_1, N_2, \cdots, N_k]$ denote the normal sample points, and $N_{k+1}, N_{k+2}, \cdots, N_{k+\beta}$ are the β abnormal sample points after N_k, i.e. $N_{k-w+\beta+1}, N_{k-w+\beta+2}, \cdots, N_{k,} N_{k+1}, N_{k+2}, \cdots, N_{k+\beta}$ are our input for ARIMA, and the number of output sample points of ARIMA equals to our sliding window size w.

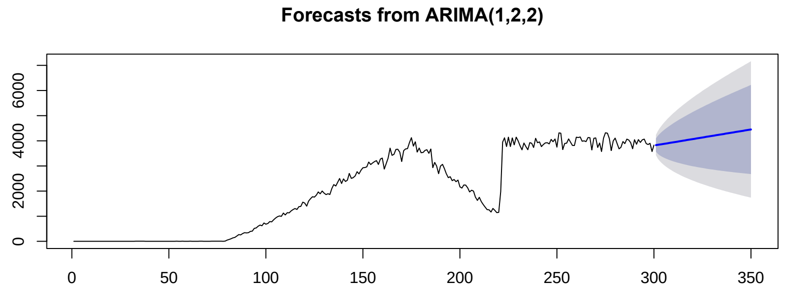

Figure 11: Forecasts of DDoS attack 2007 dataset

Within the w forecast sample points, we count the number of forecast sample points which exceeds alpha as γ. We use $\frac{\gamma}{w}$ as the probability of DDoS attack. If the probability is in $(0\%, 100\%)$, we'll move the frontier of the sliding window to the newest real sample point, and keep to sliding window size w fixed. Once the probability reaches to 100%, then we confirm that a DDoS attack starts, and pause ARIMA module for trending prediction. ARIMA module will not resume until there are consecutive β sample points' $PDRA$ value are lower than threshold value α. Now we resume the ARIMA module for trending prediction with the same sliding windows size w , i.e. input is $N_{k-w+\beta+1}, N_{k-w+\beta+2}, \cdots, N_k, N_{k+1}, N_{k+2}, \cdots, N_{k+\beta}$, and also outputs w trending points, as shown in Fig. 11.

But here we are monitoring the probability of the DDoS attack is stopped, therefore the probability is calculated as $1 - \frac{\gamma}{w}$. If the probability reaches 0%, we could confirm that the DDoS attack is stopped, and ARIMA module will be pause again until there are consecutive β abnormal points. Otherwise, we will move the frontier of the sliding window to the newest real sample point while keep the sliding window size w fixed. This process illustrated in Fig. 12. In this paper, based on the observed value, we set the sliding window size $w = 50$. The disadvantage is that, for every network, there should be a theoretically optimal sliding window size w^*. However, we currently could not find a universal optimal value for any arbitrary network. But the advantages are that, we could control the consumption of compute resources by changing the sliding window size w, and this gives us flexibility, because we could control the sliding window size w, then we could apply different sliding window size w for different time periods in a day.

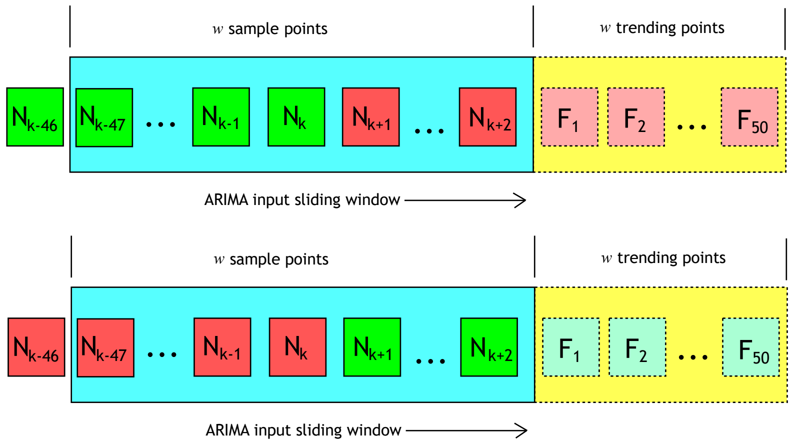

Figure 12: ARIMA sliding window

The advantages of the proposed method are listed as follow.

1) Because we requires 2 consecutive abnormal sample points to trigger the ARIMA module for trending prediction, we could effectively decrease the fluctuation caused by $PDRA$.
2) We can not only decrease the false alarm rate caused by fluctuation of single point when network flow is normal, but also decrease the missing rate caused by fluctuation of single point during the DDoS attack.
3) Furthermore, in this way we can start ARIMA module as less as possible, ARIMA module only starts and make trending predictions during the change of the network flow state, which results in less compute resources consumption.

Algorithm 2 Training

```
Input: network flow $V$ to be detected
1: while has next data do
2:     get next data $(T_i, S_i, D_i)$
3:     update $t$
4:     if $t == \Delta t$ then
5:         calucalate PDRA value according to Eq. 8
6:         if we're now in normal and current consecutive $\beta$ PDRA exceeds preset threshold $\alpha$ then
7:             get most recent $w$ PDRA values
8:             predict $w$ future PDRA values with ARIMA
9:             $\gamma \leftarrow$ the number of prediction values which exceeds preset threshold $\alpha$
10:            calculate the probability of DDoS attack $\frac{\gamma}{w}$
11:            if the probability reaches 100% then
12:                trigger DDoS alarm
13:                pause ARIMA module
14:                change network flow state to DDoS
15:            end if
16:        end if
17:        if we're now in DDoS and current consecutive $\beta$ PDRA is lower than preset threshold $\alpha$ then
18:            get most recent $w$ PDRA values
19:            predict $w$ future PDRA values with ARIMA
20:            $\gamma \leftarrow$ the number of prediction values which exceeds preset threshold $\alpha$
21:            calculate the probability of DDoS attack has stopped $1 - \frac{\gamma}{w}$
22:            if the probability reaches 0% then
23:                pause ARIMA module
24:                change network flow state to normal
25:            end if
26:        end if
27:        $t \leftarrow 0$
28:    end if
29: end while
30: return
```

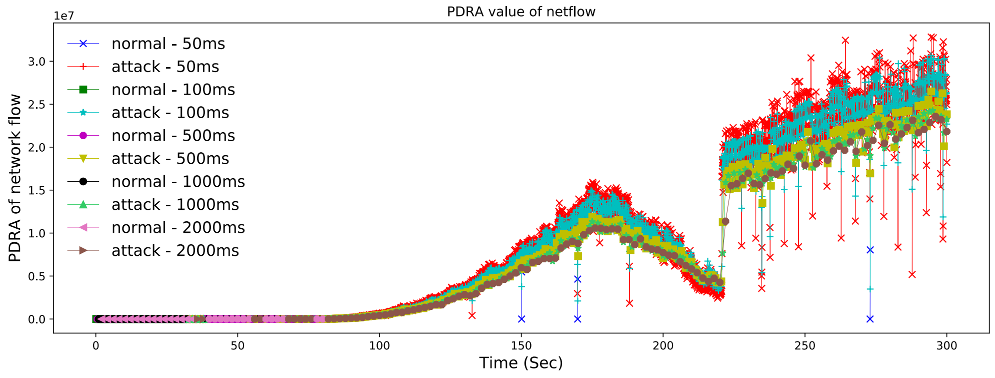

Figure 13: Original calculation of PDRA

In following experiment we conducted, after applied the ARIMA trending prediction module as we described above, the detection rate of the results was greatly improved and false alarm rate has been decreased to 0. The value of our original calculation was kept in Fig. 13, but these wrongly identified points were currently detected.

The detection algorithm is shown in Algo. 2.

4 Experiments

4.1 Dataset and assessment criterions

Our experiment was based on the dataset CAIDA DDoS Attack 2007, we trained our model with different $\Delta t = [0.05, 0.1, 0.5, 1.0, 2.0]$ seconds, and the proposed algorithm successfully distinguished normal flow and DDoS flow as shown in Fig. 14. Besides the basic DDoS detection, with the observation of our experiment, the $PDRA$ we proposed could be further used as a DDoS impact indicator.

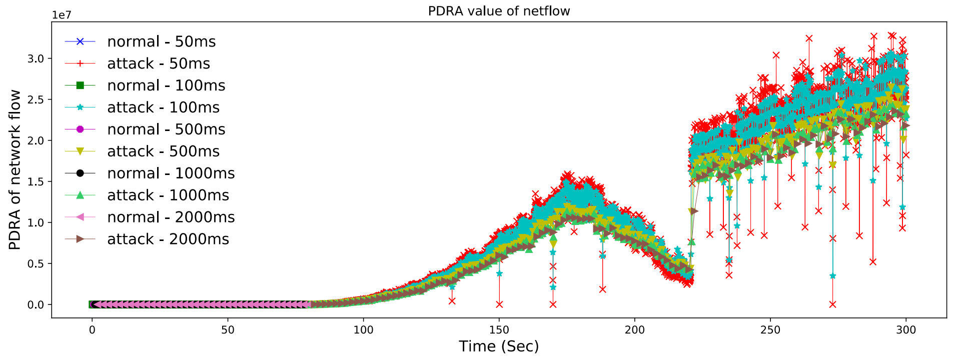

Figure 14: PDRA value of CAIDA DDoS attack 2007 dataset with different Δt

To assess the proposed algorithm, we define serval criterions. And before that, we firstly define TN as the number of correctly identified DDoS network flow samples, and FN, correspondingly, the number of samples which is normal, but incorrectly flagged DDoS network flow ones. We let TP denotes the amount of correctly identified normal users, and FP denotes the amount of samples which mistakenly flagged as normal network flow, but should be actually DDoS network flow samples.

1) **Detection Rate, DR.** This value denotes the probability of the classifier identifies actual DDoS attack flow. DR is calculated as the number of true negative samples divides the sum of both true negative and false negative samples.

$$\mathrm{DR} = \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FN}} \tag{16}$$

2) **Missing Rate, MR.** This value represents the probability of the classifier fails to identify actual DDoS attack flow. MR is calculated as the number of false negative samples divides the sum of both true negative and false negative samples.

$$\mathrm{MR} = \frac{\mathrm{FN}}{\mathrm{TN}+\mathrm{FN}} \tag{17}$$

3) **False alarm Rate, FR.** False alarm rate suggests that the probability of normal users is mistakenly flagged as attackers by the classifier. Correspondingly, it calculates as the number of false positive samples divides the sum of both true positive and false positive samples.

$$\mathrm{FR}=\frac{\mathrm{FP}}{\mathrm{TP}+\mathrm{FP}} \tag{18}$$

4.2 Experimental result

Inside the CAIDA DDoS attack 2007 dataset, it says that normal network flow starts at 13:49:36, and DDoS attack occurs at around 14:15:56. And the duration of each trace in the dataset is 5 min, so we take first 25 min' trace as our training set, i.e. training set started at 13:49:36 and ended at 14:14:36. And the rest trace is the network flow which to be detected. Rest network flow trace contains both normal network flow and DDoS attack network flow, so we can directly see the performance of the proposed method on normal network flow and DDoS attack network flow. Using the proposed method with different sample time $\Delta t = [0.05,0.1,0.5,1.0,2.0]$.

4.2.1 Prediction with traditional threshold

We firstly plot the time series of calculated $PDRA$ and judge the network status by simply a threshold of that in Fig. 13. As we can see, there are some points which was either false alarmed or failed to trigger a DDoS alarm.

Because the traditional threshold method might trigger a lot false alarms and the missing rate is also relatively high, it cannot be used as a good method in practice which requires a high standard of detection rate and false alarm rate.

4.2.2 Prediction with proposal method

And we have conducted the following experiment for comparison the choice of parameter Δt. The training set and detection set are exactly the same for each algorithm, the training set we using is started from 13:49:36 and ended at 14:14:36, and the detection set we using is started at 14:14:36 and ended at 14:19:36. The experiment result is shown in Fig. 14 and Tab. 2. Now we compare the proposed method with the previous pure threshold method by detection rate, missing rate and false alarm rate in table. DR_1,MR_1 and FR_1 are detection rate, missing rate and false alarm rate of pure threshold method. And DR_2, MR_2 and FR_2 are detection rate, missing rate and false alarm rate of proposed ARIMA trending prediction method.

Table 2: Comparison between different $\boldsymbol{\Delta t}$

Δt	DR_1	DR_2	MR_1	MR_2	FR_1	FR_2
0.05	99.50%	99.50%*	0.50%	0.00%*	0.38%	0.05%*
0.1	99.54%	100.00%*	0.45%	0.00%*	0.00%	0.00%
0.5	99.77%	100.00%*	0.23%	0.00%*	0.00%	0.00%
1.0	100.00%	100.00%	0.00%	0.00%	0.00%	0.00%
2.0	99.10%	100.00%*	0.91%	0.00%*	0.00%	0.00%

As we can see, ARIMA trending prediction module worked and the accuracy was further improved. In our analysis, applying ARIMA module to the network flow is natural, it mimics the way a human expert would probably do, that is, considering both the tendency of the most recent network flow and current network flow status, calculating the tendency and making predictions. Because our ARIMA trending prediction module takes the most recent w values of $PDRA$, which means the predictions are based on the most recent network flow status, thus it would give us a most likely future of the network flow which we are monitoring on.

It might be noticing that when we take 0.05 second as our Δt in the proposed method, the false alarm rate has been dropped from 0.38% to 0.05%, but it is still not 0. Diving into the CAIDA 2007 attack dataset, we find that in some corner situations, there are indeed less old users and relatively more new users right before the DDoS attack. And since we do not really have a clear and numerical based standard, the exact time of the early stage of a DDoS attack may be varied from different perspective views.

4.3 Comparison with previous works

To valid the method we proposed, we compared the proposed method ($\Delta t = 1.0$) with previous works, SMPM method proposed by Gu et al. [Gu, Sun and Sheng (2017)], c-SVR regression prediction method, and ARMIA method.

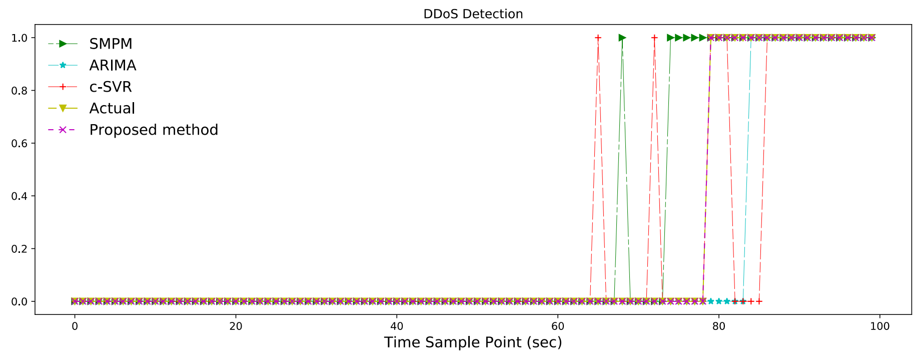

Figure 15: DDoS Detection Comparison

Let 1 denotes that DDoS attack detected, and correspondingly, 0 denotes that there is no DDoS attack. We take the first 100 seconds for detection for assessing sensitivity of these four methods, we would like to know whether a method alarms when DDoS attack happens, and if it alarms, what about the first alarm time compared to the actual DDoS attack. The results of these four methods are shown in Fig. 15.

As we can see, the detection rate of the proposed method is 100%, and the false alarm rate and missing rate are both 0% in the case, which suggests that the proposed method has a high precision rate and low false alarm rate. The detection rate of SMPM method is 100%, however, its false alarm rate is relatively high. When it comes to the *c*-SVR method, there are 4 attack samples are misidentified. The first alarm of c-SVR and SMPM methods raised before the actual DDoS attack, and for ARIMA, it delayed 5 seconds. The first alarm of proposed method raised right at the time the actual DDoS attack happened. This proves that the method we proposed has a proper sensitivity.

And the full result of the dataset we used is shown in Fig. 16. This figure suggests that, when the initial stage of a DDoS attack past, all the four methods could detect DDoS attack correctly. If a method failed at the initial DDoS attack stage, it could correctly detect the attack afterwards, because the DDoS attack flow would increase sufficiently large enough to be detected. However, we would like to know there is a DDoS attack as fast as possible, and we would also like to get rid of false alarms. The detection rate, false alarm rate and error rate are illustrated in Fig. 17.

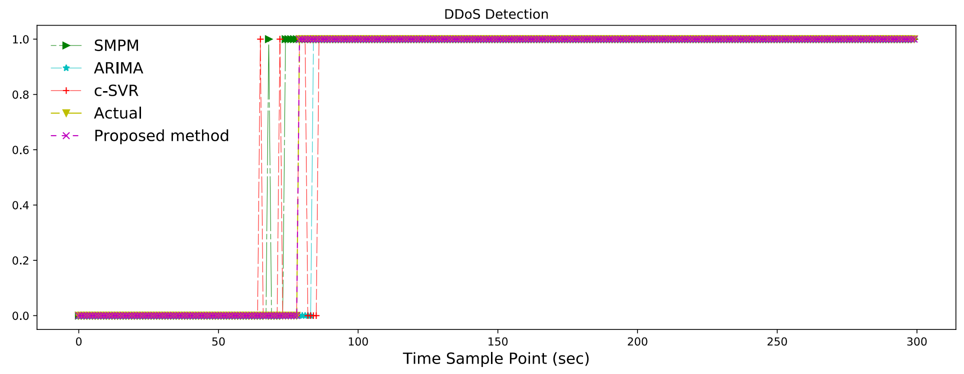

Figure 16: DDoS detection comparison, full 300 seconds

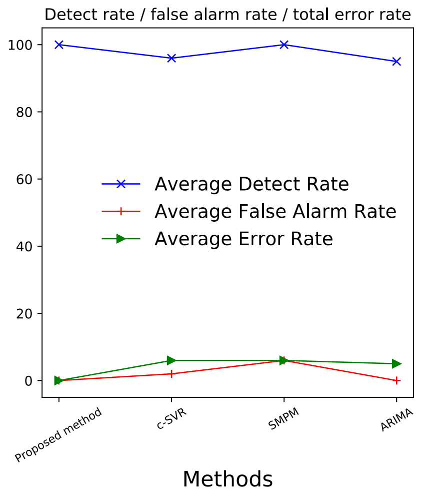

Figure 17: Detect rate, false alarm rate and error rate of the full dataset

Because our proposed method is parameterized by Δt, different Δt s are considered in the below experiment.

As illustrated in Fig. 18, compared to the c-SVR, SMPM and ARIMA methods, the proposed method using $\Delta t = [0.1,0.5,1.0]$ could successfully detect all DDoS attack samples without any false alarm. However, when $\Delta t = 0.05$, there is indeed a missing sample. To find the reason, we dive into the CAIDA DDoS 2007 dataset, and we find that if the Δt is short enough, chances are that there is just a few new users and the number of old users is approximately the same as normal times. And consequently, this would result in a relatively a small PDRA value, and because it is still in the initial stage of a DDoS attack, the predicted PDRA values given by ARIMA might be small, and leads to a missing.

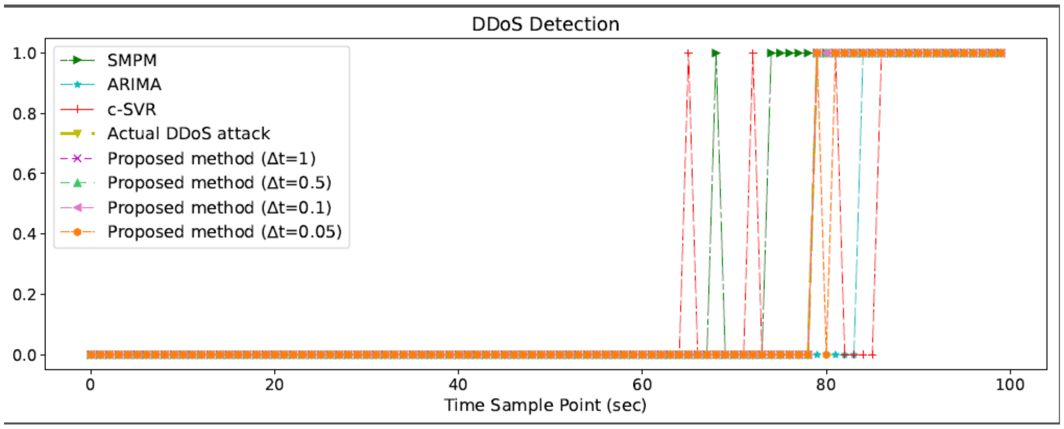

Figure 18: DDoS detection comparison with different parameters

However, such a short Δt might not be practical parameter for 2 reasons.

1) Given a short Δt would result in more computing. The prediction work is expensive, because when the network flow goes like a DDoS attack, we will active ARIMA trending prediction module for prediction. Chances are that a short Δt will trigger a lot prediction, which highly unlikely leads to a good performance.
2) If Δt is short, it is possible that there would not be that many new users/would be relatively many old users in such a short duration when DDoS attack happens. In the initial stage of a DDoS attack, even if we take past data into consideration using ARIMA, such case may still cause the detector to be failed. Besides, in real-life, the access of normal users varies, and if a shorter Δt is chosen, chances are that the fluctuation of the network flow increases. Thus the activation condition, consecutive β abnormal points, is more likely to be satisfied and cause more frequently starting of the ARIMA trending prediction module.

Taking a larger Δt would effectively avoid such situation and gives a better performance.

5 Conclusion and future direction

In this paper, we analyzed the feature of normal network flow and DDoS attack network flow. We defined the network flow abnormal index as PDRA with the percentage of old IP address, the increment of the new IP address, the ratio of new IP address to the old IP address and average accessing rate of each new IP address. Based on the $PDRA$, we have presented a method which meet the following requirements in the big data environment, to reduce the consumption of compute resources, to avoid or minimize the interference with normal network flow, and detect the DDoS attack at its early stage with high accuracy and low false alarm rate. The IP address database we designed could reach $O(1)$ time complexity, the activation mechanism of ARIMA trending prediction module could save unnecessary compute resources. Therefore, the algorithm we proposed can effectively reduce the compute resources, minimize the interference with normal network flow, we have optimized the process as much as possible, include but not limited in the process of network flow sampling, monitoring, analyzing and trending prediction.

The further work can be done in these ways, firstly, sample time Δt, we could further choose a parameter Δt which is more stable or dynamically adjust the parameter to save computing resources and elastically fit in the big data environment. And sliding window size w, the sliding window size w could be also set to dynamically according to real network status and available compute resources.

Acknowledgements: This work was supported by the National Natural Science Foundation of China [No. 61762033, 61363071, 61702539]; The National Natural Science Foundation of Hainan [No. 617048, 2018CXTD333]; Hainan University Doctor Start Fund Project [No. kyqd1328]; Hainan University Youth Fund Project [No. qnjj1444].

References

Bawany, N. Z.; Shamsi, J. A.; Salah, K. (2017): DDoS attack detection and mitigation using SDN: methods, practices, and solutions. *Arabian Journal for Science and Engineering*, vol. 42, no. 2, pp. 425-441.

Behal, S.; Kumar, K. (2017): Detection of DDoS attacks and flash events using novel information theory metrics. *Computer Networks*, vol. 116, pp. 96-110.

Boro, D.; Bhattacharyya, D. K. (2017): DyProSD: A dynamic protocol specific defense for high-rate DDoS flooding attacks. *Microsystem Technologies*, vol. 23, no. 3, pp. 593-611.

Dick, U.; Scheffer, T. (2016): Learning to control a structured-prediction decoder for detection of HTTP-layer DDoS attackers. *Machine Learning*, vol. 104, no. 2-3, pp. 385-410.

Eid, M. S. A.; Aida, H. (2017): Trustworthy DDoS defense: design, proof of concept implementation and testing. *IEICE TRANSACTIONS on Information and Systems*, vol. 100, no. 8, pp. 1738-1750.

Elejla, O. E.; Anbar, M.; Belaton, B. (2017): ICMPv6-based DoS and DDoS attacks and defense mechanisms. *IETE Technical Review*, vol. 34, no. 4, pp. 390-407.

Gu, B.; Sun, X.; Sheng, V. S. (2017): Structural minimax probability machine. *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 7, pp. 1646-1656.

Gulisano, V.; Callau-Zori, M.; Fu, Z.; Jiménez-Peris, R.; Papatriantafilou, M. et al. (2015): STONE: A streaming DDoS defense framework. *Expert Systems with Applications*, vol. 42, pp. 9620-9633.

Gurusamy, R; Subramaniam V. (2017): A machine learning approach for MRI brain tumor classification. *Computers, Materials & Continua*, vol. 53, no. 2, pp. 91-108.

Han, D.; Bi, K.; Liu, H.; Jia, J. (2017): A DDoS attack detection system based on spark framework. *Computer Science & Information Systems*, vol. 14, no. 3.

Hoque, N.; Kashyap, H.; Bhattacharyya, D. K. (2017): Real-time DDoS attack detection using FPGA. *Computer Communications*, vol. 110, pp. 48-58.

Jia, B.; Ma, Y.; Huang, X.; Lin, Z.; Sun, Y. (2016): A novel real-time DDoS attack detection mechanism based on MDRA algorithm in big data. *Mathematical Problems in Engineering*, no. 3, pp. 1-10.

Kaur, J.; Kaur K. (2017): A fuzzy approach for an IoT-based automated employee performance appraisal. *Computers, Materials & Continua*, vol. 53, no. 1, pp. 23-36.

Kim, D. H.; Lee, S. J. (2016): A method for preemptive intrusion detection and protection against DDoS attacks. *Journal of The Korea Institute of Intelligent Transportation Systems*, vol. 15, no. 2, pp. 157-167.

Latif, R.; Abbas, H.; Latif, S. (2016): Distributed denial of service (DDoS) attack detection using data mining approach in cloud-assisted wireless body area networks. *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 23, no. 1-2, pp. 24-35.

Liao, Q.; Li, H.; Kang, S.; Liu, C. (2015): Application layer DDoS attack detection using cluster with label based on sparse vector decomposition and rhythm matching. *Security and Communication Networks*, vol. 8, no. 17, pp. 3111-3120.

Lim, S.; Yang, S.; Kim, Y.; Yang, S.; Kim, H. (2015): Controller scheduling for continued SDN operation under DDoS attacks. *Electronics Letters*, vol. 51, no. 16, pp. 1259-1261.

Luo, H.; Chen, Z.; Li, J.; Vasilakos, A. V. (2017): Preventing distributed denial-of-service flooding attacks with dynamic path identifiers. *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 8, pp. 1801-1815.

Malhi, A. K.; Batra, S. (2016): Genetic-based framework for prevention of masquerade and DDoS attacks in vehicular ad-hocnetworks. *Security and Communication Networks*, vol. 9, no. 15, pp. 2612-2626.

Park, P.; Yoo, S.; Ryu, H.; Park, J.; Chung, K. H. et al. (2015): A collaborative defense mechanism against DDoS attacks for network service continuity. *ASIA LIFE SCIENCES*, pp. 93-107.

Peng, T.; Leckie, C.; Ramamohanarao, K. (2003): Protection from distributed denial of service attacks using history-based IP filtering. *IEEE International Conference on Communication*, vol. 1, no. 1, pp. 482-486.

Peng, T.; Leckie, C.; Ramamohanarao, K. (2004): *Proactively detecting distributed denial of service attacks using source IP address monitoring*. Springer Berlin Heidelberg.

Poongodi, M.; Bose, S. (2015): A novel intrusion detection system based on trust evaluation to defend against DDoS attack in MANET. *Arabian Journal for Science and Engineering*, vol. 40, no. 12, pp. 3583-3594.

Poongodi, M.; Bose, S. (2015): Stochastic model: reCAPTCHA controller based co-variance matrix analysis on frequency distribution using trust evaluation and re-eval by Aumann agreement theorem against DDoS attack in MANET. *Cluster Computing*, vol. 18, no. 4, pp. 1549-1559.

Ramanauskaitė, S.; Goranin, N.; Čenys, A.; Juknius, J. (2015): Modelling influence of Botnet features on effectiveness of DDoS attacks. *Security and Communication Networks*, vol. 8, no. 12, pp. 2090-2101.

Rashidi, B.; Fung, C.; Bertino, E. (2017): A collaborative DDoS defence framework using network function virtualization. *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 10, pp. 2483-2497.

Saied, A.; Overill, R. E.; Radzik, T. (2016): Detection of known and unknown DDoS attacks using artificial neural networks. *Neurocomputing*, vol. 172, pp. 385-393.

Schölkopf, B.; Bartlett, P. L.; Smola, A. J.; Williamson, R. C. (1999): *Shrinking the tube: a new support vector regression algorithm*. MIT Press, Cambridge, MA.

Seo, J. W.; Lee, S. J. (2016): *A study on efficient detection of network-based IP spoofing DDoS and malware-infected Systems*. Springer International Publishing.

Serwadda, A.; Phoha, V. V. (2015): When mice devour the elephants: A DDoS attack against size-based scheduling schemes in the internet. *Computers & Security*, vol. 53, pp. 31-43.

Singh, K.; Singh, P.; Kumar, K. (2017): Application layer HTTP-GET flood DDoS attacks: Research landscape and challenges. *Computers & Security*, vol. 65, pp. 344-372.

Subramanian, K.; Gunasekaran, P.; Selvaraj, M. (2015): Two layer defending mechanism against DDoS Attacks. *International Arab Journal of Information Technology*, vol. 12, no. 4.

VN Index (2017): *Cisco visual networking index: forecast and methodology 2016-2021.*

Xiao, P.; Qu, W.; Qi, H.; Li, Z. (2015): Detecting DDoS attacks against data center with correlation analysis. *Computer Communications*, vol. 67, pp. 66-74.

Yan, Q.; Gong, Q.; Deng, F. A. (2016): Detection of DDoS attacks against wireless SDN controllers based on the fuzzy synthetic evaluation decision-making model. *Adhoc & Sensor Wireless Networks*, vol. 33.

Yan, Q.; Gong, Q.; Yu, F. (2017): Effective software-defined networking controller scheduling method to mitigate DDoS attacks. *Electronics Letters*, vol. 53, no. 7, pp. 469-471.

Yan, Q.; Yu, F. R.; Gong, Q.; Li, J. (2016): Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges. *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 602-622.