Detecting DDoS Attacks against Data Center with Correlation Analysis

Peng Xiao\textsuperscript{a,b}, Wenyu Qu\textsuperscript{a,*}, Heng Qi\textsuperscript{c}, Zhiyang Li\textsuperscript{a}

\textsuperscript{a}School of Information Science and Technology, Dalian Maritime University, Dalian 116026, China
\textsuperscript{b}School of Information Science and Engineering, Dalian Polytechnic University, Dalian 116034, China
\textsuperscript{c}School of Computer Science and Technology, Dalian University of Technology, Dalian 116024, China

Abstract

Distributed Denial-of-Service (DDoS) attacks pose a great threat to the data center, and many defense mechanisms have been proposed to detect it. On one hand, many services deployed in data center can easily lead to corresponding DDoS attacks. On the other hand, attackers constantly modify their tools to bypass these existing mechanisms, and researchers in turn modify their approaches to handle new attacks. Thus, the DDoS against data center is becoming more and more complex. In this paper, we firstly analyze the correlation information of flows in data center. Secondly, we present an effective detection approach based on CKNN (K-Nearest Neighbors traffic classification with Correlation analysis) to detect DDoS attacks. The approach exploits correlation information of training data to improve the classification accuracy and reduce the overhead caused by the density of training data. Aiming at solving the huge cost, we also present a grid-based method named \textit{r}-Polling Method for reducing training data involved in the calculation. Finally, we evaluate our approach with the Internet traffic and data center traffic trace. Compared with the traditional methods, our approach is good at detecting abnormal traffic with high efficiency, low cost and wide detection range.

Key words: Data Center, DDoS, K-Nearest Neighbor, Correlation

\textsuperscript{*}Corresponding author

Email address: wenyu@dlmu.edu.cn (Wenyu Qu)

Preprint submitted to Elsevier February 11, 2015

© 2015. This manuscript version is made available under the Elsevier user license
http://www.elsevier.com/open-access/userlicense/1.0/
1. Introduction

Nowadays, data centers or large clusters of servers have been increasingly employed in many enterprises to run a variety of applications. Most of these applications are based on the TCP protocol such as web services which dominate the most part of network traffic. However, the applications of TCP are suffering from many attacks especially DDoS attacks. The mechanism of DDoS attack relies on exploiting the huge asymmetry between the Internet and a victim servers limit. When the victim server is always dealing with a large number of fake requests, the requests from legitimate users are not processed and the victim is taken off the Internet.

The key point of DDoS defenses is to detect them as soon as possible and neutralize this effect. Currently, many researchers have proposed numerous DDoS detection methods \[1, 2, 3, 4\]. However, there are many challenges to detect successfully the DDoS attack against data center:

(1) There are many different ways of perpetrating a DDoS attack. Some attacks have been handled effectively by existing defense systems, but some attacks remain unaddressed. The distinguishing between legitimate packets of normal traffic and abnormal traffic sent by compromised hosts to their victims is a hard task, because the packet header fields are modified and they look like normal ones.

(2) Data centers support a myriad of services and applications such as Web, Ftp, DNS, Hadoop, etc. Meanwhile DDoS attacks have many methods to implement such as SYN flood, DNS Query flood, HTTP flood, Low-rate DDoS, peer-to-peer-DDoS, etc. How to identify these normal flows and abnormal flows quickly and efficiently by computing once? Another challenge is that a large number of packets should be analyzed. It is difficult to improve the accuracy of detection and reduce its response time.

(3) All these methods mentioned in \[1, 2, 3, 4\] measure DDoS damage superficially and partially by measuring a single traffic parameter, such as duration, throughput, and showing divergence during the attack from the baseline case. They do not consider the distinguishing of different applications and how to select various traffic parameters. With the number of parameters increasing, the results of detection become more accurate, but the cost of computing is growing exponentially.

Motivated by these analyses, we study the problem of DDoS attack
against data center. We propose a lightweight method of detecting DDoS attacks to address the challenges. This method is implemented over a data center network by using the correlation information of training data and CKNN classification. Because of this reason, the method needs the smaller size of training data and improves the classification performance effectively.

The major contributions of this paper are:

- We propose a design to detect DDoS attacks with high efficiency and low cost, which can quickly and efficiently identify the normal flows and abnormal ones in data center by computing once.

- We provide a novel approach by using the correlation information and CKNN classification, which not only improves the classification accuracy, but also reduces the overhead significantly.

- Our experimental evaluations show that using the correlation information helps to reduce the size of training data, which in turn reduces overhead significantly and improves the accuracy of classification. Also, the classification of CKNN with grid mapping can provide fewer response time with low overhead.

The rest of the paper is organized as follows. Background and related work is discussed in Section 2. We analyze the correlation of flows in Section 3 and describe the framework design and its implementation in Section 4. A large number of experiments and results for performance evaluation are presented in Section 5. Finally, we conclude this paper in Section 6.

2. Related Work

The classification of DDoS attack or normal traffic is the weak point of DDoS defense techniques. Many classification methods have been proposed relevant to DDoS defense in the last decade, such as statistical methods, artificial intelligence, and data mining. These works can be categorized as non-machine learning methods and machine learning methods.

2.1. non-machine learning methods

DDoS attack is well known as a congestion-based attack. To detect such attacks proactively, many researches focused on the router and threshold. Carl et al. [1] introduced some DoS detection techniques such as activity
profiling, change-point detection, and wavelet-based signal analysis. They proved that it is possible to identify successfully DoS flooding attacks. Although each detector shows promise in limited testing, but not solve the detection problem. Combining various approaches with experienced network operators most likely produce the best results.

Chen et al. [2] proposed a collaborative detection mechanism where they develop a distributed change-point detection architecture using change aggregation trees. While focusing on detection rate, it is difficult for this architecture to differentiate the real attacks and normal flash crowds. As the mechanism is designed at the level and heavily relies on the normal operation of participating routers, the false positives will increase if the routers are compromised.

Xiang et al. [3] presented two information metrics such as the generalized entropy metric and the information distance metric. They used generalized entropy and information distance to detect low-rate DDoS attacks. But their solution requires to control all routers in the network and it is hard to achieve.

Thatte et al. [4] presented the parametric methods based on statistical analysis. The main limitation of the methods is that all of them only focus on IP packet headers information and are bundled with the observed variables. It is difficult to capture the application-layer attacks.

Some intelligent mechanisms [6, 7] were proposed which based on Self-Organizing Maps [5], an unsupervised artificial neural network trained with features of the traffic flow. Some well-known neural networks to detect DDoS attacks are Back Propagation (BP) [8], Radial Basis Function (RBF) [9] and Learning Vector Quantization (LVQ) [10]. They used SOM and neural network to detect abnormal traffic, but the cost of computation is higher and the efficiency of detection is lower.

Most of the previous researches have focused on the distinguishing of different attacks and proposed solutions one by one. If a data center with various applications have been attacked by many different DDoS tools, it is hard to detect for the approaches mentioned above.

2.2. machine learning methods

When machine learning theories were proposed over the last decade, the flow-based methods such as Naive Bayes [11], k-means [12], C4.5 decision tree [13], SVM [14], KNN [15] had already been used in traffic classification.

Some machine learning methods for detecting DDoS attacks on traffic have already been proposed. Oke et al. [16] proposed a generic approach
which uses multiple Bayesian classifiers to detect attacks. However, naive Bayes are based on a very strong independence assumption. Nahla Ben Amor [17] compared the performance of Naive Bayes with C4.5 decision tree, and find the good performance of Bayes with respect to existing best results performed on KDD’99. In [18], Liao and vemuri used KNN to calculate the similarity between the new process and each training process instance. Experiments show that it is excellent in attack detection, but the detector based on KNN is computationally expensive when the number of processes simultaneously increases.

The intelligent mechanism employed by our method is based on KNN classifier. In contrast to the other methods, KNN classifier has several advantages. For example, it is simple and easy to implement, and able to handle a huge number of classes. One of the main benefits is that it can support high-dimensional calculation with more features of traffic. The number of features affects the classification accuracy directly. In this point of view, The KNN classifier is more suitable for traffic classification in current complex network environment of data center.

However, the performance of the KNN is severely affected by a small size of training data which cannot accurately represent the traffic classes. Jun Zhang et al. [19] observed that the classification accuracy of the KNN decreases by approximate 20 percent when the number of training samples reduces from 100 to 10 for each class. They calculated the similarity between the correlated traffic flows which are generated by the same application and the fewer training samples. The results show that the traffic classification using few training samples can be significantly improved by this approach. However, they conduct correlation analysis using a 3-tuple heuristic \( \{dst\_ip, dst\_port, protocol\} \), which is difficult to identify a 3-tuple heuristic created by DDoS attack. This idea cannot be used in test data, but it can be used in training data especially for data center environment, which will be mentioned later.

Considering the previous KNN methods, there is commonly tradeoff between detection efficiency and cost. The high classification accuracy needs the increase of training samples, which also increases the computational overheads and memory overheads. While detecting attacks as soon as possible is important for preparing defense measures in DDoS attacks. Therefore, it is necessary to develop a method to address these drawbacks. In this paper, our work focuses on the KNN method and how to detect DDoS attacks with high efficiency and low cost in data center network.
3. Correlation Analysis of Training Data

In this section, we analyze the correlation of flows in the data center. We capture and select the flows which are generated by the same application as our training data in data center. These flows that share the same 3-tuple \{\textit{dst\_ip}, \textit{dst\_port}, \textit{protocol}\} are generated by the same application, for example, the flows created by an SYN flood attack tool stacheldraht [23] are all connecting to the same web server at TCP port 80 for a longtime. The 3-tuple heuristic about flow correlation has been considered in [19, 24, 25]. But Jun Zhang et al. [19] used it in the test data and put the correlated traffic flows into a BoF (Big of Flow). There are at least two reasons why we argue that the approach could have fatal drawbacks: (1) Obviously, it is difficult to find a BoF quickly in the traffic of a big network and the approach cannot process flows in real-time, because of the classification dealing with BoFs instead of individual flows. (2) The test flows with the same port don’t belong to the same application. For examples, web application can run on port 80, many other applications including attack tools can modify their port as 80. It is wrong that a BoF including web flows and attack flows is classified as the web application. This idea cannot be used in the test data (the unknown flows), but it can be used in the training data (the known flows) especially for data center environment.

In the paper, we use the BoF idea to model the correlation information of training data which are obtained easily by the applications running in data center. It is different from the exiting work [19], and our new research problem is how to use the group idea(BoF) in the training data, how to create the training data with correlation analysis in DDoS detection framework, and how to compute with correlation information.

The flows generated by the same application are very similar. Table 1 shows the correlation of Hadoop flows which are captured as a subset of training data from our data center. From the descriptions of data sets, we can observe that the flows generated by the same application have more similarity and correlation. Most values of the features are centralized in a smaller range segment. For example, the value of \textit{c2s\_pkts} (which means the number of packets from client to server) varies from 5 to 7764, but the samples within [5, 489] reach to 98% of the total samples.

We can easily measure the flows of different applications including normal services and abnormal DDoS attacks in data center. Based on the same application, the training data set \(D=\{d_1, d_2, ..., d_n\}\) can be divide into
Table 1: The correlation of Hadoop flows

<table>
<thead>
<tr>
<th>Feature</th>
<th>Min</th>
<th>Max</th>
<th>Centralized Range</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c2s_{pkts}$</td>
<td>5</td>
<td>7764</td>
<td>[5, 489]</td>
<td>98%</td>
</tr>
<tr>
<td>$c2s_{psmax}$</td>
<td>0</td>
<td>17376</td>
<td>[0, 868]</td>
<td>96%</td>
</tr>
<tr>
<td>$c2s_{psmin}$</td>
<td>0</td>
<td>277</td>
<td>[0, 21]</td>
<td>89%</td>
</tr>
<tr>
<td>$s2c_{pkts}$</td>
<td>3</td>
<td>6467</td>
<td>[3, 407]</td>
<td>98%</td>
</tr>
<tr>
<td>duration</td>
<td>1.3</td>
<td>33929.8</td>
<td>[1.3, 5221]</td>
<td>85%</td>
</tr>
</tbody>
</table>

many subsets which can be described as $D_{\theta_j} = \{d_1^{\theta_j}, d_2^{\theta_j}, ..., d_{m_j}^{\theta_j}\}$, $n = \sum_{j=1}^{\mid\theta\mid} m_j$

where $D_{\theta_j}$ represents a subset of training data, $\theta_j$ is the class of the subset which represents a application software, and $m_j$ is the number of flows from this application. The detailed description of the complete algorithm will be presented in section 4 later.

4. DESIGN

In this section we present a new framework which is called KNN traffic classification with correlation analysis (CKNN for short). A useful approach of grid is also proposed to reduce effectively the overhead of CKNN. We partition the training data using a spatial index named grid, which divides the space or data into a series of contiguous cells and can be assigned unique identifiers[30, 31].

4.1. System Model

Fig. 1 shows the proposed system model. Our method for DDoS attack detection consists of three modules including Flow Collector, Feature Selection and Classifier, which run on a sniffer host.

In the Flow Collector module, the system captures IP packets from the data center network and collects traffic flows by IP header inspection. A flow consists of successive IP packets having the same five-tuple \{src_ip, src_port, dst_ip, dst_port, protocol\}, and each flow can be represented by a set of statistical features, such as duration, $c2s_{pkts}$, and $c2s_{psmax}$, etc. There are also other content characteristics such as the number of failure login, the number of root access, etc.

After receiving collected flows from the Flow Collector module, the Feature Selection module extracts features that are important to DDoS flooding
attack detection, then gathers them in an $n$-tuple and passes them to the Classifier module. There are 249 kinds of TCP flows basic features mentioned in [20], and the other content characteristics can also be used to represent an attack. In this paper the Feature Selection module aims to select a subset of relevant features for building robust classification. We use the correlation-based filter selection (CFS) [28] to generate the optimal feature set. Given the features of the arrived flows, we can get the important features of the flows by the weka [26] API.

The Classifier module analyzes whether a given $n$-tuple is a legitimate traffic or a DDoS flooding attack. This classification can be made by any statistical or learning method [11, 12, 13, 14, 15]. In this work we use CKNN as the classification method. Flow correlation analysis is proposed to correlate information in the training data to improve accuracy of classification. With the number of feature increasing, the result of classification becomes more precise, but the cost of computing increases greatly. In order to reduce the computation cost, we propose a grid map mentioned later to solve the problem. By using the grid method the classifier can reduce the cost of computing significantly.
The Flow Collector module can be implemented simply by tcpdump [21] or other sniffer tools. We can use tcpdump to intercept and display all packets which are transmitted over a data center network. After getting the IP packets, we should extract features from these packets. Tstat [22] is a tool which can produce statistical data from network packets. In Feature Selection module, we can extract some useful features from the statistical data by tstat to represent each flow. The Classifier module is the key of our method. Therefore, we focus on the Classifier module instead of the other two modules in this paper.

4.2. CKNN Classification with Correlation Analysis

In machine learning, pattern recognition is the assignment of a label to a given input value. An example of pattern recognition is classification, which attempts to assign each input value to one of a given classes. Formally, the problem of traffic classification can be stated as follows:

Give a test data set \( T=\{x_1, x_2, ..., x_a\} \) which are traffic flows in data center, a training data set \( D=\{d_1, d_2, ..., d_b\} \) are generated by the same application including normal services and DDoS tools, a class set \( \theta=\{\theta_1, \theta_2, ..., \theta_c\} \) which represent the classes of the flows.

Given a particular test instance, the prior probability of each possible class is \( P(\theta|x) \). The Maximum-Likelihood classifier is:

\[
\theta^* = \arg \max_{\theta} P(\theta|x)
\] (1)

In traffic classification, the \( k \)-nearest neighbor algorithm is a method that predicts flows classes based on the \( k \) closest training examples in the feature space. A flow is classified by a majority vote of its neighbors and \( k \) is a positive integer, typically small. If \( k=1 \), then the flow is simply assigned to the single nearest neighbor. The probability of KNN is:

\[
P(\theta_j|x) = \sum_{d_i \in A(x)} sim(x,d_i) y(d_i, \theta_j)
\] (2)

\[
y(d_i, \theta_j) = \begin{cases} 
1 & d_i \in \theta_j \\
0 & d_i \notin \theta_j
\end{cases}
\] (3)

where \( x \) represents the current test instance, \( A(x) \) means finding the \( k \)-nearest neighbors, \( sim(x,d_i) = \|x - d_i\| \) is the similarity between \( x \) and \( d_i \) based on the minimum Euclidean distance criterion. \( y(d_i, \theta_j) \) is the class
attribute function, if \(d_i\) belongs to the class \(\theta_j\), the value of function is 1, otherwise 0.

According to the correlation mentioned in [19], we divide the training data set of \(D\) into many subsets by \(\theta\). The probability of KNN is adjusted as follows:

\[
P(\theta_j|x) = \frac{1}{|k|} \sum_{d_i \in A(x,\theta_j)} \text{sim}(x, d_i) \tag{4}
\]

where \(k\) also represents the \(k\) closest training examples, and \(A(x,\theta_j)\) means finding the \(k\)-nearest neighbors in \(\theta_j\).

Based on the Eq. (1) and Eq. (4), a new KNN classification approach for data center is described as follows, which is derived from pattern recognition theory.

\[
\theta^* = \arg\min_{\theta} \frac{1}{|k|} \sum_{d_i \in A(x,\theta_j)} \text{sim}(x, d_i) \tag{5}
\]

Eq. (5) can also be described as follows:

\[
\theta^* = \arg\min_{\theta} \frac{1}{|k|} \sum_{d_i \in A(x,\theta_j)} \| x - d_i \| \tag{6}
\]

By using the correlation as mentioned above, we can use the CKNN classification mechanism described by Eq. (6) to detect the possibility of an attack. In this paper, we give an example in 2D space in order to draw it easily. While flows have the feature of high-dimensional and our techniques can be easily applied to high dimensional space.

As shown in Fig. 2, we can divide training data into some subsets by the same application, which improve the classification accuracy obviously by using correlation information. After dividing training data, we can calculate distances between the test instance and the training subsets using the formula as follows:

\[
\text{avg}_j = \frac{1}{|k|} \sum_{d_i \in A(x,\theta_j)} \| x - d_i \| \tag{7}
\]

where the result is smaller, the test flow is more similar to the class of subset \(\theta_j\) according to Eq. (6).

Compared with the traditional algorithm of KNN, our algorithm can improve the classification performance effectively by incorporating correlated
information into the training data, and is affected fewer by the density of training data.

4.3. CKNN Based on Grid

The KNN algorithm is an efficient traffic categorization algorithm in recall and accuracy, but the computational overhead of KNN is directly proportional to the size of training data, and its classification speed is very low in large-scale training data. In order to keep the maximum correlation, we do not make any change to the density of training data, which increases the overhead further. Aiming at this problem, we present a grid-based method for reducing training data involved in the calculation. As discussed in [30, 31], some other spatial indices such as R tree can be applied as well, but a grid-based spatial index has the advantage that the structure of the index can be created first, and data added on an ongoing basis without requiring any change to the index structure. So the grid index is more suitable than R tree for the traffic classification in the data center. Our method maps each subset of training data into the grid, and then maps the test flow into the grid. Finally, we can calculate the distances between the test flow and the training samples in neighboring cells instead of all training data. The experiment results indicated that the method can reduce the computational overhead efficiently.
Definition 1. Given a data set $P$, which contains a set of flows. Each flow $x \in P$ is represented by $d_x = \{\mu_1^x, \mu_2^x, ..., \mu_n^x\}$. The extent of each cell on each dimension is $\lambda_n$. In this paper, we assume that the training data collected from data center is indexed by the grid index which evenly partitions the space into cells. Cell $g_x$ indicates the cell at space coordinate $(g_1^x, g_2^x, ..., g_n^x)$, which can be calculated by $g_1^x = \lfloor \mu_1^x / \lambda_1 \rfloor$, $g_2^x = \lfloor \mu_2^x / \lambda_2 \rfloor$, ..., $g_n^x = \lfloor \mu_n^x / \lambda_n \rfloor$. Clearly, any flow $x$ can fall into a cell.

Figure 3: The grid-based method of training subset

Fig. 3 shows the grid-based method of any training subset. $c2s_pkts$ means the number of packets from client to server, and $s2c_psmax$ means the max length of packets from server to client. With 2-dimensional features as an example only to draw it easily, the data space is divided into many regular cells, then the samples are scanned and mapped into corresponding cell according to the values of $\lambda_i$. For example, flows $f_1$, $f_2$ and $f_3$ are mapped into cell $g_1$. There are also many cells that have no any points in them such as $g_{16}$. When an unknown flow $X$ is captured and described as a vector $d_x = \{\mu_1^x, \mu_2^x, ..., \mu_n^x\}$, the corresponding cell coordinate $(g_1^x, g_2^x, ..., g_n^x)$ can be calculated as $g_7$ by $g_1^x = \lfloor \mu_1^x / \lambda_1 \rfloor$, $g_2^x = \lfloor \mu_2^x / \lambda_2 \rfloor$, ..., $g_n^x = \lfloor \mu_n^x / \lambda_n \rfloor$. We can...
define the value of $\lambda_i$ according to the actual ranges of feature in different dimensions, for example, as the value of $c2s\_pkts$ varies from 5 to 7764, we can choose $\lambda_i \in [5, 100]$ so that the training data will be distributed in each cell as possible.

After mapping flows into grid, we can calculate with flows only in the near cells such as gray area in Fig. 3, rather than all the training data. The KNN query problem is well-studied such as RKNN [32], moving KNN [33], parallel KNN [34], but these techniques are flat in time efficient and space efficient. Cheema M A et al. [35] presented an efficient method Circular Trip to find the near neighbors of cell. However, we find that the Circular Trip based KNN does not support the high-dimensional space, which can not be used in data center traffic classification. To find the near neighbors of cell in high-dimensional space is a hard work, we use a method named the $r$-Polling Method to solve it as Algorithm 1 shows.

**Algorithm 1: r-Polling Method**

1. INPUT: a testing sample $X$, grid index $G$ for training data $D$, the polling radius $r$, the parameter $k$ of CKNN, the values of $\lambda_i$
2. OUTPUT: the result set $\omega$
3. initialize set $\omega = \emptyset$, $\delta = Grid\_Radius$
4. map each $X_i \in X$ into the cell $g_x$ by $\lfloor X_i/\lambda_i \rfloor$
5. insert $\theta_j$ samples of $g_x$ into $\omega$
6. for (all $k$ nearest neighbors of each $\theta_j \notin \omega$ ) do
   
   for $(g_i \in G)$ do
   
   if($Dist(g_i, g_x) \leq r$)
   
   insert $\theta_j$ samples of $g_i$ into $\omega$
   
   $r = r + \delta$

11. emit $\omega$

The formula of Euclidian distance between two cells is as follows:

$$
Dist(g_i, g_j) = \sqrt{\sum_{k=1}^{n} (g_{ik} - g_{jk})^2}
$$

(8)

where $g_i, g_j$ are two cells to be measured and $n$ is the number of dimensions which is decided by the flow features. the polling radius $r$ is a variable which is gradually increasing. If $Dist(g_i, g_j) \leq r$, then the cell $g_i$ is a neighbor of
g_j, else the parameter of r continues to increase until polling end.

The whole algorithm of CKNN is outlined in Algorithm 2 as follows:

**Algorithm 2: CKNN Based on Grid Algorithm**

1. INPUT: testing data T, training data D, the parameter k of CKNN
2. OUTPUT: the result set R, flows marked with class labels
3. initialize set R = ∅, scan the data set T, D
4. for (each training sample d ∈ D) do
5.   insert d into different D_θ_j according to class θ_j
6. map each D_θ_j into grid G_θ_j
7. for (each test sample d’ ∈ T) do
8.   for (each class θ_j exist) do
9.     map d’ into G_θ_j
10.    use r-Polling Method to find the near cells S
11.   find k nearest neighbors in S
12.  calculate avg_θ_j
13.  d’.class = arg min_θ avg_θ_j
14.  insert d’ into R
15. emit R

We analyze the theoretical computation of algorithmically complexity differences between KNN and CKNN. Give a test data set \( T = \{ x_1, x_2, ..., x_\alpha \} \) with \( n \)-dimensional features samples, a training data set \( D = \{ d_1, d_2, ..., d_\beta \} \) with \( n \)-dimensional features samples, and a class set \( \theta = \{ \theta_1, \theta_2, ..., \theta_\gamma \} \) which represent the classes of the flows. The time complexity of traditional KNN algorithm is \( O(\alpha \times \beta) \) for this case. The authors [19] assumed the test data set \( T \) can be divided into \( x \) groups by the same port. Then, the method [19] needs time \( x \times O(\frac{\alpha}{x} \times \beta) = O(\alpha \times \beta) \), which has the same time complexity with the traditional KNN. Unlike the above methods, we assume each class has the same number of samples in the training data set \( D \), which can be easily generated when collecting the training data. We partition \( D \) into the \( n \)-dimensional grid space and the grid has \( e_1 \times e_2 \times ... \times e_n \) cells, where \( e_1 = \frac{\max\{d_1\}}{\lambda_1}, e_2 = \frac{\max\{d_2\}}{\lambda_2}, ..., e_n = \frac{\max\{d_n\}}{\lambda_n} \). The costs of grid are negligible compared to computing with all the training samples. If the training samples are assumed to be evenly mapped into all cells, and the r-Polling Method returns \( \varepsilon \left( 1 \leq \varepsilon \ll e_1 \times e_2 \times ... \times e_n \right) \) cells, the average case complexity of
CKNN is computed as \( \gamma \times O(\alpha \times \frac{\varepsilon}{e_1 \times e_2 \times \ldots \times e_n} \times \frac{\beta}{\gamma}) = \frac{\varepsilon}{e_1 \times e_2 \times \ldots \times e_n} O(\alpha \times \beta). \) Obviously, CKNN is more efficient than the KNN algorithm.

5. EXPERIMENTS

5.1. Testbed and data sets

In this section, we present the experimental setup and experimental results for evaluating CKNN and polling method. All algorithms mentioned in Section IV are implemented with weka [26] API, and have been performed on a Hadoop cluster which provides web service, Hadoop service, etc. The cluster consists of 12 machines running the version of Hadoop 0.20.2. One machine serves as the master node, the others are the slave nodes. Each node has 2 AMD Opteron 2212 2.00 GHz CPUs, 80 GB SCSI HDD, 8 GB of RAM, Intel 100 Mbps Ethernet Controller. All machines carry on Linux operating system of Ubuntu 10.10 server 64 bit. In the meantime, we deploy a sniffer host as a monitor of the whole network and collect the training data for our CKNN. By capturing the traffic of the known applications respectively, we can label its application class in training data as we know that the flow comes from the application. To verify the effectiveness and availability of our methods, the experiments are conducted on the following data sets:

- The wide data set, which is obtained from the wide trace [27]. The data set is from daily trace at a trans-Pacific line (150 Mbps link) and has many stochastic factors, which makes traffic classification more difficult. We use the data set to test the wide adaptability and high accuracy of CKNN, and the WWW flows which dominate the whole data set are also the main flows in data center.

- The real data set, which is a full payload traffic data set we collected at a 100 Mbps edge link of our data center located in Dalian, China over several days. We launch DDoS attacks to the data center with a series of DDoS tools and capture the traffic. The data set contains commodity traffic of our data center and abnormal traffic. Following the work in the Feature Selection module, we use the correlation-based filter selection (CFS) [28] to generate optimal feature set for the real data set.
• The KDD’99 data set, which is obtained from the KDD Cup 1999 Data [29]. This data set is used for The Third International Knowledge Discovery and Data Mining Tools Competition, which contains a standard set of data to be audited and a wide variety of intrusions simulated in a military network environment. We select a subset from the data set to test our predictive model capable of distinguishing between attacks and normal connections.

The traffic of these data sets contain several different protocols: about 85% is TCP, about 10% is UDP, the remaining is ICMP. The differences of three data sets are shown in Table 2. For example, the wide data set has 22,011 testing flows which consists of 6 different flow types, and each flow type has a corresponding class in data set.

Table 2: The features of each data set

<table>
<thead>
<tr>
<th>Data set</th>
<th>The number of samples</th>
<th>The number of classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>wide</td>
<td>22011</td>
<td>6</td>
</tr>
<tr>
<td>real</td>
<td>106538</td>
<td>8</td>
</tr>
<tr>
<td>KDD’99</td>
<td>233283</td>
<td>11</td>
</tr>
</tbody>
</table>

5.2. Performance evaluation

In this section, we conduct a group of experiments to verify our methods. This group of experiments contains: (1) We evaluate the performance of CKNN with different $k$ and the number of dimensions. (2) We compare the performance of CKNN with KNN. (3) We evaluate the influence of correlation analysis. (4) We evaluate the accuracy of CKNN with more attack types.

(1) The influence of $k$ and the number of dimensions

To evaluate the influence of $k$ and the number of dimensions, we test the performance of CKNN on the wide data set when $k$ varies from 2 to 100. We mainly evaluate three aspects: accuracy, time, and flows/sec. We study the accuracy of CKNN with different $k$ and the number of dimensions. As shown in Figure 4(a), the accuracy is greatly affected by $k$. The value of accuracy is higher when $k$ varies from 5 to 6. Furthermore, the value of accuracy is sharply down when $k$ is more than 20, which is more obvious for
the 6-dimensional data set. As is to be expected, the increase of dimension can improve accuracy, but it leads to a longer response time, seeing Fig. 4(b). When \( k \) is selected as 6, the accuracy of the 6-dimensional data set only improves 2.9% than the 4-dimensional data set, but the classification time is 2 times longer than the 4-dimensional data set. Fig. 4(c) shows the speed of CKNN processing flows per second. From it we can see, the speed is sharply down with the increasing of \( k \). When \( k \) is selected as 6 and the number of feature is 4, the speed of CKNN reaches 314 flows/sec, which is enough to filter our data center traffic.

(2) The comparisons of CKNN and KNN

In this part, We compare the performance of CKNN with the traditional
KNN method. Fig. 5 shows their comparisons on the 4-dimensional wide data set. The classification time of CKNN is fewer than half of the traditional KNN. When $k = 6$, CKNN takes 70 seconds to classify the data set, while KNN takes 138 seconds to do it. As shown in Fig. 5(a), the results demonstrate that the grid method can significantly improve the performance of CKNN. Compared with the traditional KNN, it is seen that CKNN not only has better performance but also has better accuracy. It is clear that the correlation information improves the classification accuracy of CKNN as shown in Fig. 5(b).

(3) The influence of correlation analysis

In order to evaluate the influence of correlation analysis, we test the performance of CKNN with different amounts of training samples on the wide data set. Fig. 6 shows the accuracy of CKNN for each $k$ using 100 and 1000 training samples per-class. We can see that the classification accuracy is low due to very few training samples. Especially in the case of $k = 100$, the gap between the 1000 training samples per-class and the 100 training
samples per-class can be up to 30%. Therefore, few training samples severely impact the classification performance. CKNN has much higher accuracy with 1000 training samples per-class than 100 training samples per-class. The results show the method without correlation analysis can reach the target of high accuracy, but the size of training data set is so big that it needs more overhead.

Based on above results with the *wide* data set, we use correlation analysis to build a smaller supervised training data than the *wide* data set and carry out some experiments on the *real* data set. The correlation is commonly present in the Internet and data center network, so we use stacheldraht tool to generate the traffic for the DDoS flooding attack.

Table 3: The detection results with 500 training samples per-class

<table>
<thead>
<tr>
<th></th>
<th>6-feature accuracy</th>
<th>4-feature accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>http</code></td>
<td>94.1%</td>
<td>91.3%</td>
</tr>
<tr>
<td><code>BT</code></td>
<td>90.6%</td>
<td>88.4%</td>
</tr>
<tr>
<td><code>DNS</code></td>
<td>94.6%</td>
<td>91.2%</td>
</tr>
<tr>
<td><code>HADOOP</code></td>
<td>96.7%</td>
<td>93.3%</td>
</tr>
<tr>
<td><code>DOS</code></td>
<td>94.8%</td>
<td>91.9%</td>
</tr>
</tbody>
</table>

Table 3 presents the results obtained from the experiments on the *real* data set, the $k$ value is 6 and 500 training samples per-class are selected. It compares the 6-feature and the 4-feature in regard to the different class, and the value of DOS is the mean of some common DOS types. The class with high correlation is easy to be detected such as Hadoop, SYN flood. With the correlation analysis, our approach can reach high accuracy only with 500 training samples per-class, but the *wide* data set need 1000 training samples per-class as Fig. 6 shows. The experiment results show that the combination of correlation analysis and supervised training data can improve the detection performance.

In order to compare our method to other approaches with few training data, we built experiments on the *real* data set. For CKNN, the $k$ value is 6 and the 4-dimensional data set is selected. Fig. 7 shows the overall performance of four competing methods on the *real* data set. The results show that CKNN outperforms other classification methods including AVG-NN [19]. With the same numbers of training data, CKNN improves the detection performance effectively.
The accuracy of CKNN with more attack types

The KDD’99 data set has 22 classes such as back, buffer_overflow, ftp_write, guess_passwd, imap, ipsweep, neptune, nmap, portsweep, smurf, spy, teardrop, warezclient, warezmaster. We select 11 common classes including normal traffic to test the wide adaptability and accuracy of CKNN with more attack types, the statistical results are reported in Table 4. From the results, we observe that the classification of CKNN is widely adaptable in real attack traffic.

![Figure 7: The comparison of methods on the real data set](image)

Table 4: The detection results with more attack types

<table>
<thead>
<tr>
<th>Attack types</th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOS(Neptune, back, smurf, teardrop, pod)</td>
<td>96.3%</td>
</tr>
<tr>
<td>probe(portsweep, satan, ipsweep)</td>
<td>95.4%</td>
</tr>
<tr>
<td>r21(nmap, warezclient)</td>
<td>95.6%</td>
</tr>
</tbody>
</table>
6. CONCLUSION

With more and more applications are deployed in data center, the detection of DDoS attacks is becoming more and more difficult. In this paper, we present a method named CKNN to solve this problem. Although there have been lots of studies on the detection of DDoS attacks, they mostly focused on the distinguishing of different attacks and found a different method to detect a particular attack. As our method is based on flows, we are thereby able to detect the existing attacks by examining flow features only. With the correlation analysis, we can find the hidden relations of training data from data center, which can improve the classification accuracy and is not affected by the density of training data. To reduce the overhead of KNN, we map the training data into grid. The testing samples are only calculated with the training samples in neighboring cells instead of all the training data by using r-Polling Method, which can reduce the overhead of CKNN efficiently. Furthermore, the CKNN method is affected less by the density of training data which directly influences the efficiency and precision of traditional KNN classifier. Otherwise, to keep the maximum correlation, we do not make any change to the original density of the training data.

In order to evaluate this detection method, we experiment with the Internet, data center traffic trace and the KDD’99 data set. The results show CKNN is a suitable way to classify network traffic, because of its competence in DDoS detection even with a high rate of noisy data signals. Moreover, our method is easy to be implemented since it uses correlation information of training data. There are some issues worthy of future research, in theory CKNN is suitable to detect any network attack flow. In the future work, we expect to build a bigger training data including more network attack types, which can help construct a powerful defense system of data center against attacks.

7. Acknowledgement

This work is supported by the National Science Foundation for Distinguished Young Scholars of China (Grant No. 61225010), the State Key Program of National Natural Science of China (Grant No. 61432002), National Nature Science Foundation of China (Nos. 61370199, 61370198 and 61402069), the Prospective Research Project on Future Networks from Jiangsu Future Networks Innovation Institute, and the Fundamental Research Funds for the Central Universities (Nos. 3132014325, 3132013335).
References


Peng Xiao is a Ph.D. student in School of Information Science and Technology, Dalian Maritime University, China. He is also a lecturer in School of Information Science and Engineering, Dalian Polytechnic University. His research interests include traffic classification and Software Defined Network.

Wenyu Qu received the bachelor’s and master’s degrees from Dalian University of Technology, China in 1994 and 1997, and the Ph.D. degree from Japan Advanced Institute of Science and Technology, Japan in 2006. Now she is a professor at the School of Information Science and Technology, Dalian Maritime University, China. She was a assistant professor in Dalian University of Technology, China from 1997 to 2003. Her research interests include cloud computing, computer networks and information retrieval. She has published more than 80 technical papers in international journals and conferences. She is on the committee board for a couple of international conferences.

Heng Qi is a lecture at the School of Computer Science and Technology, Dalian University of Technology, China. He got bachelor’s and master’s degrees from Hunan University in 2004 and Dalian University of Technology in 2006, respectively. He was a software engineer in GlobalLogic-3CIS from 2006 to 2008. Then he got his doctorate degree from Dalian University of Technology in 2012. His research interests include multimedia computing, multimedia applications and mobile cloud computing. He has published more than 20 technical papers in international journals and conferences.

Zhiyang Li is a lecturer in School of Information Science and Technology, Dalian Maritime University, China. He received his Ph.D. degree from Dalian University of Technology, China, in 2011. His research areas include Information Retrieval and Computer Vision.