Machine-Learning-Based Online Distributed Denial-of-Service Attack Detection Using Spark Streaming

Baojun Zhou
Department of Computer Science
University of Tsukuba, Japan
Email: zhoubaojun@osdp.cs.tsukuba.ac.jp

Jie Li
Faculty of Engineering,
Information and Systems
University of Tsukuba, Japan
Email: lijie@cs.tsukuba.ac.jp

Jinsong Wu
Department of Electrical Engineering
University of Chile, Chile
Email: wujs@ieee.org

Song Guo
Department of Computing
Hong Kong Polytechnic University, China
Email: cssongguo@comp.polyu.edu.hk

Yu Gu
School of Computer and Information
Hefei University, China
Email: yugu.bruce@ieee.org

Zhetao Li
College of Information Engineering
Xiangtan University, China
Email: liztchina@gmail.com

***Abstract*—In order to cope with the increasing number of cyber attacks, network operators must monitor the whole network situations in real time. Traditional network monitoring method that usually works on a single machine, however, is no longer suitable for the huge traffic data nowadays due to its poor processing ability. In this paper, we propose a machine-learning-based online Internet traffic monitoring system using Spark Streaming, a stream-processing-based big data framework, to detect DDoS attacks in real time. The system consists of three parts, *collector*, *messaging system* and *stream processor*. We use a correlation-based feature selection method and choose 4 most necessary network features in our machine-learning-based DDoS detection algorithm. We verify the result of feature selection method by a comparative experiment and compare the detection accuracy of 3 machine learning methods - Naïve Bayes, Logistic Regression and Decision Tree. Finally, we conduct experiments in a cluster with the standalone mode, showing that our system can detect 3 typical DDoS attacks - TCP flooding, UDP flooding and ICMP flooding at the accuracy of more than 99.3%. It also shows the system performs well even for large Internet traffic.**

I. Introduction

Network security has always been a very important issue. [1], [2] Traditionally, Internet traffic monitoring is executed on a high-performance central server [3]. However, due to the limitation of computing ability, the central server cannot cope with a large volume of data, such as huge Internet traffic, in a short time. For example, when a distributed denial-of-service (DDoS) attack occurs, we need to monitor the huge amount of Internet traffic, which is a tough task for the single server. Some monitoring methods use packet sampling to reduce the amount of input data, which, however, makes the results inaccurate [4]. Moreover, the single server makes the system vulnerable to failures. If the server crashes, we cannot recover it quickly without affecting the ongoing task [5].

In this paper, we focus on a typical network attack, DDoS, and propose a machine-learning-based online Internet monitoring system based on Spark Streaming, a big data platform that (i) can process huge traffic data efficiently so that we can monitor network status in real time, and (ii) is robust enough that a failure will not abort the whole monitoring process.

Big data platforms, such as Hadoop and Spark, provide us an efficient way to process huge data. For example, the MapReduce model and its open-source version Hadoop [6] have been widely adopted in the big-data analytics community for their simplicity and ease-of-programming [7]. However, the intermediate data of Hadoop are stored on disk (usually with a low performance in I/O), which may lead to a dramatic performance degradation for algorithms that require lots of iterations. To improve the performance of Hadoop, the in-memory computing methods, such as Apache Spark [8], have been proposed. The intermediate data in Spark are stored in RDD (Resilient Distributed Datasets) that are cached in memory, so it can process data much faster than Hadoop [9].

Some offline Internet monitoring systems [3], [10]–[12] have used the big data platforms to improve their processing efficiency. However, only few works [13]–[16] have focused on the online network monitoring.

Both Hadoop and Spark are based on batch-processing which is suitable for offline data analysis. Batch-processing is applied to processing large datasets, where operations on multiple data items can be batched for efficiency [17]. Batch-processing requires all the input data to be accessible when calculation begins so that all the data can be processed together. Online Internet traffic monitoring is more like a stream analytics problem where the input comes as an unbounded sequence of data. Although MapReduce does not support stream processing, it can partially handle streams using a technique known as micro-batching. The idea is to treat the stream as a sequence of small batch chunks of data. On small intervals, the incoming stream is packed to a chunk of data and are delivered to the batch system to be processed [18]. Spark, for example, has provided a library, Spark Streaming, to support this technique. There are also some other platforms that are inherently designed for big data streams, such as Apache Storm and S4, where data is processed through several computing nodes. Each node can process one or more input stream(s) and also generate a set of output streams. Data will be processed immediately when they arrive.

Attack detection with machine learning requires some network characteristics (such as average flow length, protocol ratio, and so on) to predict the attacks. Those parameters are often measured statistically on a set of packets in a period, and thus micro-batching may be a better choice. Here, we use Spark Streaming in our monitoring system.

We list 7 network features that are closely related to the DDoS detection in Section IV. However, in order to build an accurate and well-performed machine learning classifier, we firstly need to use feature selections to filter the features. In this paper, we use the correlation-based feature selection technique [19] to filter the features.

Our contributions are as follows.

- We propose a distributed architecture for the online DDoS detection system.
- We use the correlation-based feature selection method and finally choose 4 most necessary features in our machine-learning-based DDoS detection system. We also compare the accuracy of 3 machine learning algorithms.
- We have conducted typical experiments showing that the proposed system is feasible and efficient.

The rest of this paper is organized as follows. In section II, we introduce some related works on real-time Internet monitoring. In section III, we describe the architecture of our proposed monitoring system and introduce the tools we use. In section IV, we elaborate the algorithms and show how DDoS detection problem can be solved by the parallel method in our system. In section V, we test the performance of the proposed system and show the experiment results. Finally, we draw the conclusions in section VI.

II. Related Work

Some related works have been conducted in online Internet monitoring system using Spark. Gupta, et al. [13] used Spark Streaming to analyze network in real time. They show three network monitoring applications that can be expressed as streaming analytics problems - reflection attack monitoring, application performance analysis, and port scan detection. They made use of the programmable switches such as OpenFlow switches, to extract only the traffic that is of interest, thus reducing the data that needed to be processed. However, their system is not feasible for networks that use non-programmable switches. In this paper, we propose an Internet traffic measurement and monitoring system that works on both programmable and non-programmable switches. Karimi, et al. [14] proposed a distributed network traffic feature extraction method for real-time IDS (Intrusion Detection System) with Spark. They used a collector component to capture packets from the switch and extract required packet headers information. These headers are saved into CSV files, separated by the time window. Spark then reads data from CSV files periodically within a small-time window to make it nearly real-time. However, the periodic writing and reading of files will degrade the performance of Spark for the online Internet traffic monitoring system. Our system uses Spark Streaming to cope with the stream directly so that achieves a higher speed.

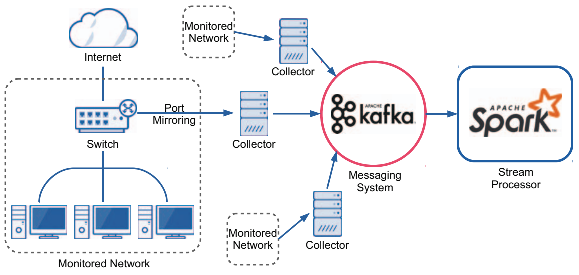

Figure 1: The architecture of the proposed online DDoS detection system.

III. The Architecture of Proposed System

As illustrated by Figure 1, the proposed online DDoS detection system consists of 3 components: *collector*, *messaging system*, and *stream processor*. A *collector* is a device which is used to capture packets from the network. It captures all the inbound and outbound packets from one switch through port mirroring. There may be multiple *collector*s in the system if we need to capture packets from multiple switches. The *stream processor* is the core component of our system, which processes the input data transmitted from *collectors*. In order to reduce the amount of input data, the *collector* pre-processes the captured packets firstly and only sends necessary protocol header data to the *stream processor*. Moreover, we use a *messaging system* as a bridge to help transmit the data from the *collectors* to the *stream processor*.

To realize the *collector* component, we use `Jpcap`, which is an open-source Java library, to capture packets and extract required packet header data (such as time stamp, TCP port number, etc.) efficiently. In order to reduce the number of captured packets, we set a filter (protocol, destination IP address, etc.) for `Jpcap` to let it only capture the packet we are interested in.

We use Kafka to build up our *messaging system*. Kafka is a high performance distributed messaging system for streams of records [20]: 2 million writes per second on three cheap machines. The *collector*s publish required packet header data as messages on Kafka using a common topic, then the *stream processor* subscribes to the topic and obtains the message stream from Kafka.

The *stream processor* component is a cluster running Spark Streaming. It processes the packet information collected by *collector*s and shows monitoring results to operators. Similar to Spark program for batch processing, whose logic is expressed by transformations on RDD (Resilient Distributed

Table 1: Some useful transformation APIs in Spark Streaming

Transformation	*Meaning*
map()	Map each element in the source stream to a new value.
flatMap()	Similar to map(), but each element can be mapped to 0 or more output items.
mapValues()	Map the value of each key-value pair without change the key.
reduce()	Aggregate each 2 elements in the source stream to 1 new element.
reduceByKey()	Aggregate 2 key-value pairs with the same key to a new key-value pair.
groupByKey()	Group all key-value pairs with the same key together.
countByValue()	Count the frequency of each element, and return a key-value pair stream whose key is the element, value is the count.
join()	Join two key-value pair streams (K, V) and (K, W) together, return a new stream of (K, ⟨V, W⟩) pairs with all pairs of elements for each key.

Datasets), logic in Spark Streaming is expressed by transformations on DStream (discretized stream, which is internally a sequence of RDDs) [21]. Spark Streaming has provided a lot of transformation APIs, such as `flatMap()` which maps each input item in source DStream to 0 or more output items, `groupByKey()` which can group key-value pairs together according to their keys, etc. Some useful transformation APIs are shown in Table 1 [21].

IV. DDoS Attack Detection

DDoS is a relatively simple, yet very powerful technique to attack Internet resources as well as system resources. The attacker uses distributed multiple agents to consume a large number of system resources within a short time and make the target system unavailable for legitimate users [22]. Typical DDoS attacks are UDP flood, ICMP flood, TCP (HTTP) flood, etc. Many DDoS attack detection methods have been proposed, such as machine-learning-based method and statistical-based method, etc. Among them, the machine-learning-based methods have the following advantages: (i) Adaptability. We can update their strategies with new information. (ii) High detection rate for known attacks. In our system, we use machine learning methods to detect DDoS attacks.

A. Detection Features

Detection features are used by machine learning model to predict whether the network is under attack. In our system, we should choose the detection features that can be calculated in parallel so that the computation can be accelerated by the Spark Streaming.

We choose 7 detection features.

1) Average Length of IP Flow (L_{ave_flow}): IP flow is a set of packets with the same 5-element-tuple (source IP address, source port, destination IP address, destination port, and protocol). We consider three kinds of IP flow-TCP, UDP, and ICMP. We use it as a detection feature, since IP flow in DDoS attack stream usually has a shorter length (the number of packets in certain flow).

$$L_{ave\_flow} = \frac{\sum \text{Length of IP flow}}{\sum \text{IP flows}}$$

2) The Ratio of TCP Protocol (R_t), Ratio of UDP Protocol (R_u) and Ratio of ICMP Protocol (R_i): Since TCP, UDP and ICMP are used in DDoS frequently, their proportions in traffic stream can imply the existence of DDoS.

$$R_t = \frac{\sum \text{TCP packets}}{\sum \text{IP packets}}$$

$$R_u = \frac{\sum \text{UDP packets}}{\sum \text{IP packets}}$$

$$R_i = \frac{\sum \text{ICMP packets}}{\sum \text{IP packets}}$$

3) Entropy of Protocols (E_p): The normal network traffic usually has a stationary ratio of TCP, UDP and ICMP packets. While in attack stream, whole bandwidth is flooded by attack packets, so the protocol entropy (randomities) would decrease.

$$E_p = -R_t \ln R_t - R_u \ln R_u - R_i \ln R_i$$

4) Incoming and Outgoing Ratio of IP Packets (R_{io}): In DDoS attack flow, because the attack packets are incoming, R_{io} would ascend quickly.

$$R_{io} = \frac{\sum \text{Incoming IP packets}}{\sum \text{Outgoing IP packets}}$$

5) Source IP Address Number and Destination IP Address Number Ratio (R_{sd}): In the normal case, a packet from IP *A* to *B* usually has a response from IP *B* to *A*, so the total number of source IP address should near to the number of destination IP, i.e. the ratio is close to 1. However, when the DDoS attack occurs, the attacker generates a lot of fake source IP addresses and the target fails to respond to all the incoming packet, so R_{sd} would ascend.

$$R_{sd} = \frac{\sum \text{Source IP addresses}}{\sum \text{Destination IP addresses}}$$

These 7 features are very typical ones that some of them have also been used in some DDoS detection systems [22], [23].

B. Correlation-Based Feature Selection

Feature selection is very important for machine-learning-based methods. A good set of features lead to a high detection accuracy. On the contrary, using irrelevant or redundant features will not only decrease the accuracy but also waste computing resources [24]. In this section, we introduce a correlation-based feature selection method proposed in [19].

The feature selection is based on two principles [19]: (i) A feature is useful if it is correlated with or predictive of the class; otherwise it is irrelevant. (ii) Redundant features should be eliminated. That is, we should choose a set of features that have the highest relevance with the classification but the least relevance with each other. For a feature set S, we obtain the evaluation metric M_S.

$$M_S = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}} \quad (1)$$

where k is the number of features in the feature set, r_{cf} is the mean feature-class correlation and r_{ff} is the mean feature-feature inter-correlation.

The correlation between 2 random variables *X* and *Y* can be expressed by the *information gain*. Information gain is the reduction in uncertainty about the value of *Y*, after observing values of *X*, as shown in (2).

$$\begin{aligned} gain =& H(Y) - H(Y|X) \\ =& H(X) + H(Y) - H(X,Y) \end{aligned} \tag{2}$$

where

$$\begin{aligned} H(Y) &= -\sum_{y \in Y} p(y) \log_2 p(y) \\ H(Y|X) &= -\sum_{x \in X} p(x) \sum_{y \in Y} p(y|x) \log_2 p(y|x) \\ H(X,Y) &= -\sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2 p(x,y) \end{aligned}$$

Information gain can be used to measure the correlation between 2 variables. But it is biased in favor of features with more values. Here, we can use *symmetric uncertainty* to normalize the value [25], as shown in (3).

$$\text{symmetric uncertainty} = 2 \times \frac{gain}{H(X) + H(Y)} \tag{3}$$

For each subset of the 7 proposed features, we calculate the evaluation metric M_S. The subset with the highest metric will be chosen. The experiment result is shown in Section V.

C. DDoS Attack Detection with Proposed System

We introduce how DDoS attack detection can be conducted by our system in this section.

1) Detection Feature Extraction: Firstly, we set a filter for *collectors* and let them capture all IP packets from the Internet. For TCP and UDP segment, we need its protocol, source and destination IP addresses, and ports information; For the packet of other protocols, we need its protocol, source and destination IP addresses information.

In every small time interval (1 second), we compute the needed network features. In order to calculate the ratio of each protocol (R_t, R_u, R_i) and entropy of protocols (E_p), we firstly use Spark Streaming to map each packet into an array of numbers. For TCP segment, we generate array [1, 0, 0, 1]; For UDP segment, we generate array [0, 1, 0, 1]; For ICMP packet, we generate array [0, 0, 1, 1]; For other packets, we generate array [0, 0, 0, 1]. Then, we collect and reduce all the arrays we generated by summing up the numbers in each position separately. After the reducing, we obtain an array of [the number of TCP packets, the number of UDP packets, the number of ICMP packets, the number of all packets]. Then, R_t, R_u, R_i and E_p are calculated. In the same way, we can calculate the incoming/outgoing ratio (R_{io}) and the source IP count/destination IP count ratio (R_{sd}) easily.

The average length of IP flow (L_{ave_flow}) is a bit hard calculated. Firstly, we map each packet as a tuple of data. For TCP and UDP segment, we generate a tuple of ⟨protocol, source IP address, source port, destination IP address, destination port⟩; For ICMP packet, we generate a tuple of ⟨protocol, source IP address, destination IP address⟩; For other packets, we do not generate anything. Secondly, use `countByValue()` method to group all the packets with the same tuple together and get their counts. According to the definition of flow length, the number of packets with the same tuple equals to the length of each IP flow. Thirdly, we map the result to an array [length of flow, 1], and use `reduce()` to sum up all the arrays. Then, we obtain an array of [total length of all flows, the number of flows]. Finally, we calculate the average length of IP flow (L_{ave_flow}).

2) Machine Learning Model Training and Test: Many machine learning algorithms can also be benefited from the MapReduce computing model in Spark Streaming [26]. For example, In the Naïve Bayes, we have to estimate the probability $P(f_i = a_i|C_j)$ and $P(C_j)$ from the training data, where f_i is the feature, a_i is a value, and C_j is the classification label. In order to do so, we need to sum over $f_i = a_i$ for each C_j label in the training data. The counting process can be accelerated through distributed computing in Spark Streaming.

We invoke three machine learning methods, Naïve Bayes, Decision Tree, and Logistic Regression, in our DDoS attack detection framework. We use `MLlib`, a High-quality machine learning library in Spark, to implement the machine learning algorithms.

V. Performance

We use 8 servers in Amazon Web Services (AWS) to implement the proposed online DDoS detection system. The configurations of them are shown as below.

1) Collector: 2 machines. Model t2.micro, 1GB memory.

2) Messaging System: 1 machine. Model r4.large, High-frequency Intel Xeon E5-2686 v4 (Broadwell) Processors, 15.25GB memory. Bandwidth up to 10 Gbps.

3) Stream Processor: 5 machines. Model c4.large, High-frequency Intel Xeon E5-2666 v3 (Haswell) processors, 3.75GB memory. Bandwidth 500Mbps.

In order to test the performance of our system, we implement a special *collector* which reads packets from packet capture files (.pcap) and generates a packet stream at a specified rate.

The batch interval is set to 1 seconds, i.e. the Spark Streaming will start a computation every 1 seconds and all segments information in this time interval will be collected and processed as a batch task.

In order to train the DDoS attack detection model of our system, we prepared two kinds of packets data set, stored in the packet capture files.

- Normal packets: Captured from the campus network.
- DDoS packets: Generated by `bonesi`, a DDoS botnet simulator. We simulate 3 kinds of DDoS attacks, ICMP, UDP, and TCP (HTTP) flooding.

We consider 4 different categories, normal, light, medium, and heavy to indicate the strength of the DDoS attacks. In order to generate the data set with different attack densities, we let the special *collector* read packets from both normal and DDoS packet capture files and merge DDoS attack packets into normal traffic with different ratios.

For each category, the *stream processor* collects all the packets from the simulated traffic and works out the network features in every one second time interval. We label the feature set of each time sample with its category and use it as the training data. Finally, we train our machine learning models using the generated training data set.

The details of these 4 categories are shown in Table 2. Each category has totally 300 training data (i.e. 300 time samples). For the light, medium and heavy categories, there are 100 data set for each kind of attack (i.e. TCP, UDP and ICMP flooding) respectively.

A. Feature selection

After we obtain the sample data, we can use the correlation-based feature selection to filter the features.

Firstly, we have to discredit the features so that we can calculate the information gain. We use a simple discretization method that divides the range of each feature into 50 equal-sized bins.

The feature subsets with the top evaluation metric are shown in Table 3. We find that the subset {Incoming and outgoing ratio (R_{io}), Ratio of ICMP protocol (R_i), Average length of IP flow (L_{ave_flow}), Source IP address number and destination IP address number ratio (R_{sd})} has the highest evaluation metric, so we use these 4 features in our DDoS detection method.

B. Performance evaluaton

We study the DDoS attack detection performance of the proposed system in terms of both accuracy, time cost, and robustness.

1) Accuracy: We assume that there are several categories A, B, C and so on in a data set and we want to use a machine learning algorithm to classify them. The true positive ratio (TPR) of class A is the hit ratio that a data is predicted as A by the algorithm correctly, and the false positive ratio (FPR) of class A is the false alarm ratio that a data is not A but predicted as A by the algorithm incorrectly. We can use TPR and FPR to indicate the accuracy of machine learning algorithms.

We generate additional 300 data for each category as test data set. The TPR and FPR for each machine learning algorithms are shown in Table 4. Moreover, their average TPRs for all categories are 94.1%, 96.5% and 99.3%, respectively. We can find that decision tree has the best detection accuracy.

In order to verify the effect of the feature selection, we compare the results with the accuracy of using all 7 features, shown in Table 5. We can see that the accuracy of Naïve Bayes is improved after the number of detection features is reduced to 4, while the accuracy of Logistic Regression decreases. The accuracy of Decision Tree is almost unchanged. Taking into account the reduction in computing costs due to the reduction in the number of detection features, we think the feature selection is necessary.

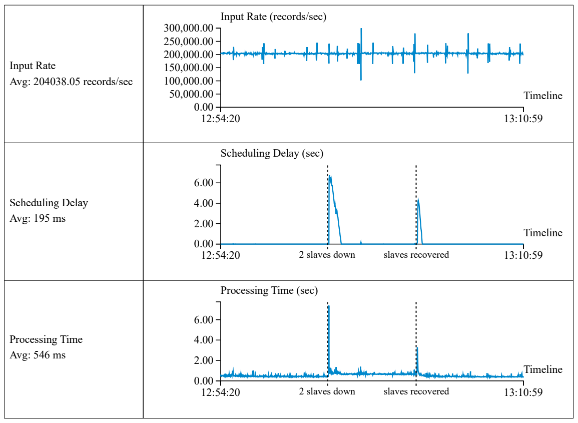

Figure 2: System performance changes when 2 slaves are down.

2) Time Cost: We generate 200,000 packets per second using the special *collector* as the input stream. We test for 300 seconds, the average time consumptions (the time consumption of network features extraction plus the time consumption of prediction) in 1 second time interval for the Naïve Bayes, Decision Tree, and Logistic Regression algorithms are 427ms, 405ms and 411ms, respectively. Considering bias of computer's operating environment, we think that they cost almost the same time. We increase the packets rate and find that our system can deal with nearly 500,000 packets per second.

3) Robustness: Our system has a better robustness, comparing to the traditional network monitoring systems, which execute the processing on a single server and get aborted when the server downs. In a cluster that is set up with Spark standalone mode, the jobs will be distributed to and completed by several machines, say slaves. When a slave broke down, the others can continue its remain jobs, without aborting the whole processing.

We have conducted a typical experiment to study its robustness of our proposed system. We shut down 2 slaves manually when the program is running and restart them lately. The system status is shown in Figure 2. When the slaves go down, their works are retransmitted to other slaves, and we can find a jitter in scheduling delay and processing time. After the slave is recovered, master has to retransmit the jobs to it, so there is also a jitter in the scheduling delay. Although the failure and recovery of a slave have an impact on the system performance, we can see that the system returns to normal very quickly and the whole job is not aborted. This experiment results show that the system has a good robustness.

VI. Conclusions and Future Works

With the growth of Internet traffic, traditional network monitoring methods working on a single machine are no longer suitable. Existing approaches have taken advantages of big data frameworks to improve the processing efficiency, but they mainly focused on offline data analysis. In this paper, we have proposed a machine-learning-based online DDoS attack detection system using Spark Streaming. We

Table 2: Details of each category.

Categories	*Attack packets ratio (Normal : DDoS)*	*Number of time samples*
Normal	Only normal packets	300 time samples
Light	2:1	100 time samples for TCP flood+100 time samples for UDP flood+100 time samples for ICMP flood
Medium	1:1	100 time samples for TCP flood+100 time samples for UDP flood+100 time samples for ICMP flood
Heavy	1:10	100 time samples for TCP flood+100 time samples for UDP flood+100 time samples for ICMP flood

Table 3: Part of feature subsets, listed in descending order of evaluation metric M_S.

Subset of features (S)	*Evaluation metric (M_S)*
$\{R_{io}, R_i, L_{ave\_flow}, R_{sd}\}$	0.608
$\{R_{io}, L_{ave\_flow}\}$	0.590
$\{R_{io}, L_{ave\_flow}, R_{sd}\}$	0.586
$\{R_{io}, R_{sd}\}$	0.574
$\{R_{io}, R_i, L_{ave\_flow}\}$	0.559
$\{R_{io}, R_i, R_{sd}\}$	0.548
$\ldots$	$\ldots$
$\{R_{io}, R_t, R_u, R_i, E_p, L_{ave\_flow}, R_{sd}\}$	0.500
$\ldots$	$\ldots$

Table 4: True positive ratio (TPR) and false positive ratio (FPR) of using 4 features.

Categories	*Naïve Bayes*		*Logistic Regression*		*Decision Tree*	
	TPR(%)	FPR(%)	TPR(%)	FPR(%)	TPR(%)	FPR(%)
Normal	100	0	100	0	100	0
Light	87.3	3.3	91	1.6	98.7	0.4
Medium	89	4	95	3	98.7	0.4
Heavy	100	0.5	100	0	100	0

Table 5: True positive ratio (TPR) and false positive ratio (FPR) of using all 7 features.

Categories	*Naïve Bayes*		*Logistic Regression*		*Decision Tree*	
	TPR(%)	FPR(%)	TPR(%)	FPR(%)	TPR(%)	FPR(%)
Normal	100	0	100	0	100	0
Light	79.3	8.1	98	0.2	98.6	0.7
Medium	74.6	6.7	98.3	0.5	97.7	0.4
Heavy	100	0.4	100	0.4	100	0

have shown that DDoS attack monitoring could be treated as a stream analytics problem and coped with streaming processing platform. Extensive experiment results show that our system can achieve a good performance and robustness.

As an ongoing project, we are investigating the DDoS attack detection accuracy only using the simulated DDoS attack packets. In the future, we will study the detection accuracy with real attack packets. We will also compare the performance of our system with some related works and the traditional single server systems from both scalability and reliability.

References

[1] J. Li, R. Li, and J. Kato, "Future trust management framework for mobile ad hoc networks," *IEEE Communications Magazine*, vol. 46, no. 4, 2008.

[2] H. Lu, J. Li, and M. Guizani, "Secure and efficient data transmission for cluster-based wireless sensor networks," *IEEE transactions on parallel and distributed systems*, vol. 25, no. 3, pp. 750–761, 2014.

[3] Y. Lee, W. Kang, and H. Son, "An internet traffic analysis method with mapreduce," in *Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP*, pp. 357–361, IEEE, 2010.

[4] D. Brauckhoff, B. Tellenbach, A. Wagner, M. May, and A. Lakhina, "Impact of packet sampling on anomaly detection metrics," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pp. 159–164, ACM, 2006.

[5] Y.-y. Qiao, Z.-m. Lei, Y. Lun, and M.-j. GUO, "Offline traffic analysis system based on hadoop," *The Journal of China Universities of Posts and Telecommunications*, vol. 20, no. 5, pp. 97–103, 2013.

[6] "Hadoop." http://hadoop.apache.org/. Accessed: 2017-10-29.

[7] K. Kambatla, G. Kollias, V. Kumar, and A. Grama, "Trends in big data analytics," *Journal of Parallel and Distributed Computing*, vol. 74, no. 7, pp. 2561–2573, 2014.

[8] "Apache spark." http://spark.apache.org/. Accessed: 2017-10-29.

[9] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets.," *HotCloud*, vol. 10, no. 10-10, p. 95, 2010.

[10] J. Liu, F. Liu, and N. Ansari, "Monitoring and analyzing big traffic data of a large-scale cellular network with hadoop," *IEEE network*, vol. 28, no. 4, pp. 32–39, 2014.

[11] Y. Lee and Y. Lee, "Toward scalable internet traffic measurement and analysis with hadoop," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 1, pp. 5–13, 2013.

[12] Z. Chen, G. Xu, V. Mahalingam, L. Ge, J. Nguyen, W. Yu, and C. Lu, "A cloud computing based network monitoring and threat detection system for critical infrastructures," *Big Data Research*, vol. 3, pp. 10–23, 2016.

[13] A. Gupta, R. Birkner, M. Canini, N. Feamster, C. Mac-Stoker, and W. Willinger, "Network monitoring as a streaming analytics problem," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, pp. 106–112, ACM, 2016.

[14] A. M. Karimi, Q. Niyaz, W. Sun, A. Y. Javaid, and V. K. Devabhaktuni, "Distributed network traffic feature extraction for a real-time ids," in *Electro Information Technology (EIT), 2016 IEEE International Conference on*, pp. 0522–0526, IEEE, 2016.

[15] B. Zhou, J. Li, S. Guo, J. Wu, Y. Hu, and L. Zhu, "Online internet traffic measurement and monitoring using spark streaming," in *Proc. IEEE GlobeCom 2017*, IEEE, 2017.

[16] B. Zhou, J. Li, X. Wang, Y. Gu, L. Xu, Y. Hu, and L. Zhu, "Online internet traffic monitoring system using spark streaming," *Journal of Big Data Mining and Analytics*, vol. 1, no. 1, 2018.

[17] C. P. Chen and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on big data," *Information Sciences*, vol. 275, pp. 314–347, 2014.

[18] S. Shahrivari, "Beyond batch processing: towards real-time and streaming big data," *Computers*, vol. 3, no. 4, pp. 117–129, 2014.

[19] M. A. Hall, "Correlation-based feature selection for machine learning," 1999.

[20] "Kafka performance." https://engineering.linkedin.com/kafka/benchmarking-apache-kafka-2-million-writes-second-three-cheap-machines. Accessed: 2017-10-29.

[21] "Spark streaming." http://spark.apache.org/docs/latest/streaming-programming-guide.html. Accessed: 2017-10-29.

[22] M. Suresh and R. Anitha, "Evaluating machine learning algorithms for detecting ddos attacks," *Advances in Network Security and Applications*, pp. 441–452, 2011.

[23] T. Xu, D. He, and Y. Luo, "Ddos attack detection based on rlt features," in *Computational Intelligence and Security, 2007 International Conference on*, pp. 697–701, IEEE, 2007.

[24] T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.

[25] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[26] C.-T. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. Bradski, K. Olukotun, and A. Y. Ng, "Map-reduce for machine learning on multicore," in *Advances in neural information processing systems*, pp. 281–288, 2007.