An Efficient Elephant Flow Detection with Cost-Sensitive in SDN

Peng Xiao[*, †], Wenyu Qu[*], Heng Qi [‡], Yujie Xu[*], Zhiyang Li[*]
[*]College of Information Science and Technology, Dalian Maritime University, Dalian 116026, China
[†]School of Information Science and Engineering, Dalian Polytechnic University, Dalian 116034, China
[‡]School of Computer Science and Technology, Dalian University of Technology, Dalian 116024, China
Email: wenyu@dlmu.edu.cn

***Abstract*—The software defined networking (SDN) allows separating control and data plane, which provides better network management and higher utilization for data center network. Among these topical applications in SDN, such as traffic engineering, QoS and network management, there is significant interest on classifying the flows and predict future traffic. Classification plays an important role in SDN, especially for elephant flow detection. However, how to efficiently detect all kinds of flows with low cost still remains a challenge task in current researches. To address this issue, in this paper, we propose to introduce cost-sensitive learning method to define a real-time elephant flow detection strategy and the subsequent metric in flow detection. Then we apply our strategy to train and evaluate cost-sensitive decision trees in SDN. Extensive experiments on different settings and data sets have been performed, showing that our strategy is good at detecting elephant flow with high detection rates and low overhead.**

Keywords—SDN; Elephant Flow; Cost-Sensitive; C4.5;

I. INTRODUCTION

Recently, software defined networking (SDN) [1] has increasingly employed in data center. SDN has caused a paradigm shift in communication networks and provides a globally optimal management of network resources and a flow-level control of network traffic. The most popular example of SDN is OpenFlow [2], its controller defines packet handling rules for switches. Thus, the problem of real-time elephant flow detection is becoming increasingly important for SDN management.

The centralized network control has also been considered in data center [3]. SDN is a natural match for data center because the applications running in data center are already managed by a central entity, which can provide higher utilization and better network management for data center. SDN is equally attractive to manage WAN networks, where logically centralized control simplifies traffic engineering problems and packets routing [4]. Detecting elephant flows is important to construct appropriate forwarding policy for various types of flows.

Currently, many researchers have studied the elephant flow problem. By measuring the flows of 10 data centers, Benson et al. [5] found that 80% of the flows (or mice flows) are smaller than 10KB in size and the most bytes are carried in the top 10% large flows (or elephant flows). Thus, it is not necessary for controller to operate all flows and direct their traffic. In the meantime, the misclassifications cost of elephant flows are more serious than one of mice flows.

To reduce work load on control plane and improve the efficiency of traffic engineering, an elephant flow detection approach had been proposed by AR Curtis et al. [6,7]. This approach is designed to allow aggressive use of wild-carded OpenFlow rules. But TCAM is a valuable resource in switches which can only afford 1,500 wild-carded rules in OpenFlow, which is difficult to scale up to operate the rapidly changing traffic. It will need other hosts to meet the demands of flow detection because of the limited processing capacity of TCAM. Some researchers [8,9] proposed their detection systems, which are pre-configured with a fixed value. The detection will cause a lot of false positive and negative errors because they do not consider the dynamically changing traffic characteristics.

All these methodologies mentioned in [6-9] have fast detecting speed but not high accuracy of detection. To improve the detection accuracy, the machine learning methods such as Naive Bayes [10], k-means [11], C4.5 decision tree [12], SVM [13], KNN [14] had already been used in traffic classification. All these approaches classify the flows by measuring the flow features, which are a set of statistical values from the flow beginning to end, such as duration. However, the arrival and departure of flows are very fast in data center [5], and it requires the controller to spend 10ms to allocate resource for every new flow. Although the above flow-based methods improve the classification accuracy markedly by using machine learning methods, they cannot detect flows in real-time.

The key point of elephant flow detection is to detect them as soon as quickly and efficiently. Currently, many researchers have proposed many detection methods. However, there some bottlenecks that control plane is difficult to detect flows for the traffic engineering:

(1) The conflict between the accuracy of classification and real-time is irreconcilable. In data center networks, the arrival and departure of flows are very fast, and the classification accuracy for elephant flow must be high. The methods based on statistical thresholds have higher speed in real-time, but lower accuracy. Conversely, the classifications based on flow features have higher accuracy, but it can not be in real-time. This problem needs to be optimized, especially for data center network.

INISCom 2015, March 02-04, Tokyo, Japan

DOI 10.4108/icst.iniscom.2015.258274

(2) The majority of existing detection methods are designed to minimize the number of errors. Nevertheless, the elephant flow detection often require classifiers that minimize the misclassifications costs. In SDN flow prediction, failing to detect an elephant flow can have more serious consequences than mice flow.

Motivated by above analysis, we propose a real-time elephant flow detection strategy and define some metrics for flow detection that considers cost-sensitive. The major contributions of this paper are:

(1) We propose a real-time elephant flow detection system which provides more high accuracy for the online detection. By analyzing the relationship between statistical thresholds and flow features in data center, we propose a two-stage elephant flow detection strategy. The strategy can detect elephant flows as soon as quickly and efficiently.

(2) We define the metrics of the flow detection regarding not only accuracy, but also cost-sensitive. Our aim in this work is to study and implement the appropriate strategy for learning and testing cost-sensitive decision trees. By considering cost-sensitive, the accuracy of elephant flow detection is much higher.

(3) We evaluate the performance of this detection system on real trace of the Internet and data center with Mininet [15]. The performance evaluations show that our approach can significantly improve the accuracy of elephant flow detection.

The rest of the paper is organized as follows. In Section Ⅱ, we give an overview on the elephant detection system. We propose an real-time elephant flow detection strategy aiming at improving the traffic measurement performance. We discuss cost-sensitive learning and define the metrics in Section Ⅲ. Our work is implemented and evaluated in Section Ⅳ. Finally, we conclude in Section Ⅴ.

II. SYSTEM DESIGN

In this section we briefly introduce the framework of elephant flow detection for SDN, and propose a useful real-time elephant flow detection strategy to effectively solve the conflict mentioned above.

A. System Model

Fig. 1 shows the proposed system model. Our method for elephant flow detection consists of three modules including Flow Collector, Real-time Detection Strategy and Classifier, which run on the controller host.

In the Flow Collector module, the system captures IP packets from the SDN network and collects traffic flows by IP header inspection. A flow consists of 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_psmax and c2s_pkts, etc. Once based on the whole flow, the classification cannot detect flows online because it needs to wait until the flow finishes itself. To avoid this problem, we propose the flow collect method based on the first time series.

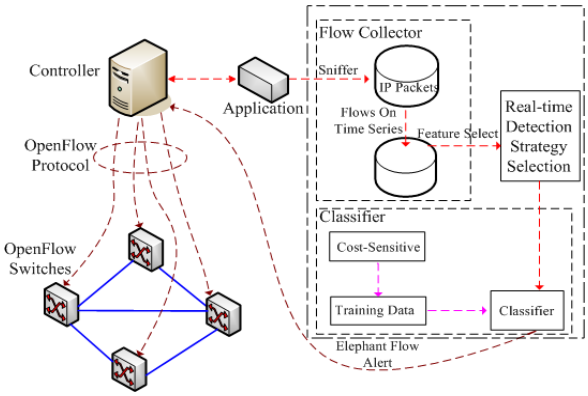

Fig. 1 System model

After receiving the flows on time series from the above module, the two-stage Detection Strategy module extracts features that are important to SDN flow detection. There are 249 kinds of TCP flows basic features mentioned in [16], and the other deep packet inspection (DPI) [17] characteristics can also be used to represent the flows in SDN. In this system the Real-time Detection Strategy module aims to select a n-tuple features to build a robust classification.

The Classifier module analyzes whether a given n-tuple is an elephant flow or not. This classification can be made by any statistical or learning method [10-14]. In this work we use C4.5 decision tree as the detection method. Cost-sensitive is proposed to improve accuracy of elephant flow detection, which is significant in traffic engineering for SDN.

The Flow Collector module can be implemented simply by wireshark [18] or other sniffer tools such as tcpdump. We use wireshark to intercept and display all packets which are transmitted over a SDN network. After obtaining the IP packets, we extract features on time series from these packets. In Real-time Detection Strategy module, we extract some useful features from the statistical data to represent each flow. Wireshark and Tstat [19] can be used in the above two modules, and the Classifier module is the key of our method. Therefore, our work focuses on using the cost-sensitive decision trees to detect elephant flows in this paper.

B. Elephant Flow Detection Strategy

Many services and applications are deployed in data center, such as web, ftp, DNS, Hadoop, VMware, etc. For these applications, some are elephant flows such as Hadoop MapReduce, VMware migration, and others are mice flows in most cases. Currently, some researchers have proposed detection methods based on port, protocol or head packet detection. However, there are many challenges to detect elephant flows successfully. For example, the web services of 80 port can transport both http request and video file. So it is difficult to detect a web flow only based on head packet, though it enjoys advantages in the speed of detection. The flow-based classification has high detection accuracy, while it needs to wait until the flow finishes itself. The arrival and departure of flows are very fast in data center, and it is difficult to detect flow by flow-based classification in real-

time. Therefore, it is necessary to develop a detection strategy to address these drawbacks for data center flow detection.

In this section, we present the two-stage elephant flow detection strategy as shown in Fig. 2. The strategy can detect elephant flows quickly and efficiently, on the basis of statistical thresholds and flow features on time series.

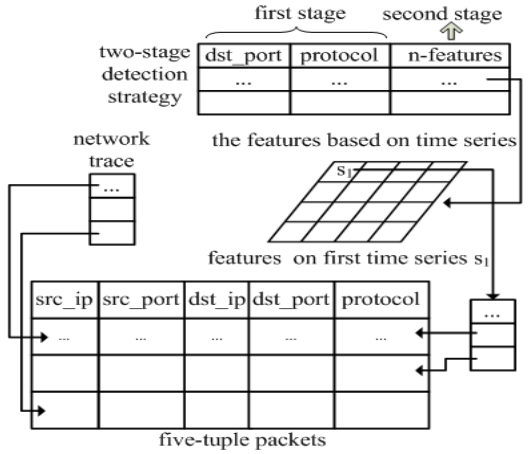

Fig. 2 Two-stage flow detection strategy

The flow is the large-scale data on time series. We adopt head packet measurement in the first stage of detection to distinguish predicting flows from mice flows by the threshold of dst_port, protocol, etc. Then, these suspicious elephant flows are sent to the second stage to improve the accuracy. In the second stage, n-features are provided by statistics information from the flow on first time series such as c2s_pkts, s2c_psmax, etc. Following the work in the Feature Selection module, we use the correlation-based filter (CFS) [20] to generate optimal feature set for the data set.

In the following sections, we introduce cost-sensitive to optimize the classification accuracy problem by using decision trees. To test our methods, we design the optimization targets for cost-sensitive detection.

III. COST-SENSITIVE ANALYSIS

In recent years, cost-sensitive has become one of the most popular machine learning algorithms [21]. The success of such algorithms heavily depends on the choice of the costs. From the analysis regarding classification speed, we tend to select C4.5 decision trees as the classification method. In contrast to the other machine learning methods, C4.5 classifier has several advantages. For example, it is simple and easy to implement, and able to handle a huge number of packets with high speed, which has been widely used on network switch. Aiming to understand the benefits of cost-sensitive, we have evaluated our elephant flow detection using cost-sensitive.

A. Problem Formulation

In machine learning, classification is the assignment of a label to a given input value. The SDN elephant flow detection is also the same process that assigns each input value to one of a given flow classes. Formally, the problem of elephant flow detection can be stated as follows:

Given a test data set $T = \{f_1, f_2, ..., f_n\}$ which are traffic flows in data center, a training data set $D = \{d_1, d_2, ..., d_n\}$ are generated by machine learning including elephant flows and mice flows, a class set $\theta = \{\theta_1, ..., \theta_i, ..., \theta_j\}$ which represent the classes of the flows.

Given a particular test instance, the prior probability of each possible class is $P(\theta | f_x)$. Mathematically, the class of flow is:

$$\theta = \arg\max_{\theta} P(\theta | f_x) \quad (1)$$

Given a specification of costs for correct and incorrect flow predictions, an example should be predicted to have the class that leads to the lowest cost. Let the (i, j) entry in a cost matrix C be the cost of predicting class θ_i when the true class is θ_j. If $i = j$ then the prediction is correct, while if $i \neq j$ the prediction is incorrect. Thus, the optimal prediction for a flow f_x is the class θ_i that minimizes

$$L(f_x, i) = \sum_j P(j | f_x) C(i, j) \quad (2)$$

For instance, failing to detect an elephant flow can have more serious consequences than mice flow in SDN flow prediction. Based on Eq. (2), we prefer to rely on the costs rather than the probability $P(j | f_x)$.

B. Costs Matrix and Optimal Decisions

Given an example in the two-class case, a cost matrix C always has the following structure:

	actual negative	actual positive
predict negative	C(0,0)=C_{00}	C(0,1)=C_{01}
predict positive	C(1,0)=C_{10}	C(1,1)=C_{11}

In the two-class case of the elephant and mice flow, the optimal prediction is elephant flow if and only if the expected cost of this prediction is less than or equal to the expected cost of predicting mice flow:

$$P(j=0|f_x)C_{10} + P(j=1|f_x)C_{11} \leq P(j=0|f_x)C_{00} + P(j=1|f_x)C_{01} \quad (3)$$

Given $p = P(j=1|f_x)$, Eq. (3) is equivalent to:

$$(1-p)C_{10} + pC_{11} \leq (1-p)C_{00} + pC_{01} \quad (4)$$

When the equation is in fact an equality, p is the threshold for making optimal decisions.

$$p = \frac{C_{10} - C_{00}}{C_{10} - C_{00} + C_{01} - C_{11}} \quad (5)$$

Eq. (5) shows that the elephant and mice flow cost matrix has essentially only one degree of freedom from a decision-making perspective, though it has two degrees of freedom from a matrix perspective.

In the following section, we evaluate the performance of our methods with cost-sensitive, and create a set of advanced testing scenarios to verify it.

IV. EXPERIMENTS

In this section, we present our experimental setup and describe the experimental results of our methods compared with the others.

A. Testbed and data sets

All methods mentioned above are implemented with weka [22] API, and have been performed on our data center which provides Floodlight, Mininet, Hadoop service, web service, etc. Two Hadoop clusters consist of 32 machines running in the data center. All machines carry on linux operating system of Ubuntu server 64 bit. In the meantime, we deploy a sniffer host as a monitor of the whole network and collect the traffic data for our methods. To verify the effectiveness and availability of our methods, the experiments are conducted on the following data sets:

(1) The wide data set, which is obtained from the wide trace [23]. 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 the cost-sensitive decision trees. In this data set we select WWW flows as the important flows because the WWW flows dominate the Internet flows.

(2) The data center data set, which is a full payload traffic data set we collected at a 100 Mbps edge link of our data center mentioned above over several days. We launch cloud computer applications to the Hadoop clusters with large data and capture the traffic. The data set contains elephant traffic and mice traffic of our data center, such as Hadoop, DNS, WWW, etc.

B. Performance evaluation

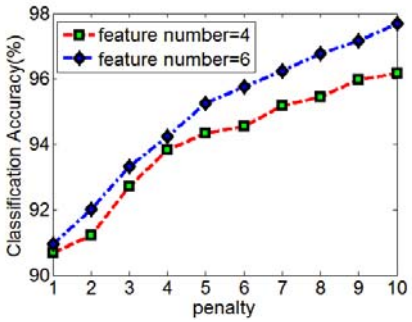

Fig. 3 The accuracy with different penalty

In this section, we conduct a group of experiments to verify our methods. This group of experiments contains: (1) We evaluate the performance of cost-sensitive decision trees with cost-sensitive analysis and the number of dimensions. (2) We compare the different data sets with cost-sensitive. (3) We compare the performance of our methods with others.

To evaluate the influence of the cost penalty and the feature dimensions, we test the performance of our methods on the wide data set. We select 1000 training samples per-class as the training data. The penalty of misclassification cost is set to an integer from 1 to 10. As shown in Fig. 3, the classification accuracy is greatly affected by penalty and the feature dimensions. Furthermore, the value of accuracy is sharply up when the penalty varies from 1 to 5, 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.

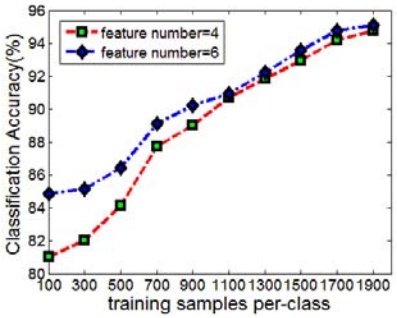

Fig. 4 The accuracy with different training samples

In order to evaluate the influence of cost-sensitive, we test the performance of C4.5 with different amounts of training samples on the wide data set. Fig. 4 shows the accuracy of C4.5 without cost-sensitive using different training samples. We can see that few training samples severely impact the classification performance. The results show C4.5 without cost-sensitive analysis can reach the target of high accuracy, but the size of training data set is so big that it needs more overhead.

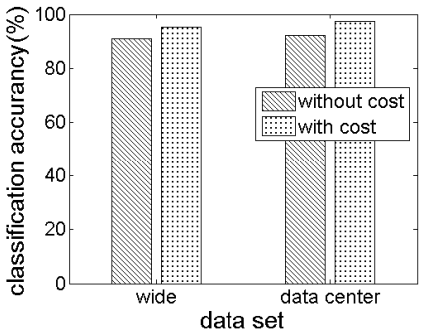

Fig. 5 The accuracy with cost and data set

Fig. 5 shows the results with cost obtained from the experiments on the two data set. Based on the above results with the wide data set, we use 10% of each data set as the training data and select 6-dimensional features. Moreover, the penalty value is 5. From the results, we observe that the classification accuracy with cost-sensitive has generally improved in different data sets, especially in the data center data set. The data center traffic is different from the Internet traffic. For the data center data set, 80% of the flows are smaller than 10KB in size and the most bytes are carried in the top 10% elephant flows. It is clear that cost-sensitive analysis is suitable for the data center data set.

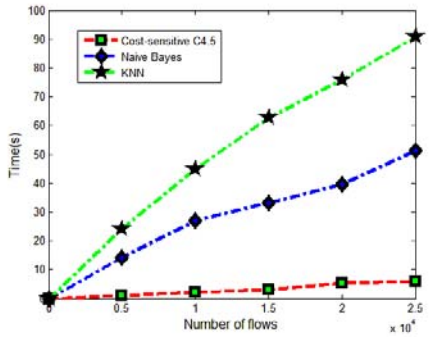

Fig. 6 The comparison of methods on the data center data set

To compare our methods with the other approaches, we built experiments on the data center data set. Fig. 6 shows the delay time of three competing methods. In our system, the detection delay includes head packet delay in the first stage and n-features classification delay in the second stage. In the first stage of head packet detection, Flow Collector module mentioned above is responsible for distinguish and mark the suspicious elephant flows for further detection in the second stage. The head packet delay takes few milliseconds for first stage, which can be ignored. In this experiment, we evaluate the delay performance without the head packet delay, because this delay is very short and head packet is only a small part of the whole flow. Our detection strategy can detect an elephant flows in few tens of milliseconds. The average duration time of elephant flows is about several seconds. As shown in Fig. 6, our methods only spend 2 seconds in filtering 10,000 flows, which is enough to filter and detect the elephant flows in data center.

V. CONCLUSION

In this paper, we showed how to detect the elephant flow with cost-sensitive in SDN. Our approach is based on cost-sensitive decision trees by using our elephant flow detection strategy. To maximize the detection rates and minimize the misclassifications costs of elephant flows, we presented the metrics of the cost-sensitive decision trees. The ideas and mechanisms are illustrated by using the Internet and data center trace. We conducted experiments under many different settings and data sets. Finally the experiment results show our methods are good at solving the elephant flow detection problem. In the future work, we expect to expand our analysis to the SDN network deployed in large enterprises. We also want to find a method to determine the suitable cost matrix automatically for decision trees.

Acknowledgment

This work is supported by National Nature Science Foundation of China (Nos. 61370199, 61370198, 61432002 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

[1] Feamster, Nick, Jennifer Rexford, and Ellen Zegura. "The road to SDN: an intellectual history of programmable networks." ACM SIGCOMM Computer Communication Review 44.2 (2014): 87-98.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," SIGCOMM CCR, vol. 38, no. 2, 2008.

[3] Al-Fares, Mohammad, et al. "Hedera: Dynamic Flow Scheduling for Data Center Networks." NSDI. Vol. 10. 2010.

[4] Jain, Sushant, et al. "B4: Experience with a globally-deployed software defined WAN." Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM. ACM, 2013.

[5] Benson, Theophilus, Aditya Akella, and David A. Maltz. "Network traffic characteristics of data centers in the wild." Proceedings of the 10th ACM SIGCOMM conference on Internet measurement. ACM, 2010.

[6] Curtis, Andrew R., et al. "Devoflow: scaling flow management for high-performance networks." ACM SIGCOMM Computer Communication Review. Vol. 41. No. 4. ACM, 2011.

[7] Curtis, Andrew R., Wonho Kim, and Praveen Yalagandula. "Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection." INFOCOM, 2011 Proceedings IEEE. IEEE, 2011.

[8] Mori, Tatsuya, et al. "Identifying elephant flows through periodically sampled packets." Proceedings of the 4th ACM SIGCOMM conference on Internet measurement. ACM, 2004.

[9] Kumar, Abhishek, et al. "Data streaming algorithms for efficient and accurate estimation of flow size distribution." ACM SIGMETRICS Performance Evaluation Review 32.1 (2004): 177-188.

[10] A. W. Moore and D. Zuev, "Internet traffic classification using Bayesian analysis techniques." in SIGMETRICS. ACM, 2005, pp. 50–60.

[11] J. Erman, M. F. Arlitt, and A. Mahanti, "Traffic classification using clustering algorithms." in Proceedings of the ACM SIGCOMM Workshop on Mining Network Data. ACM, 2006, pp. 281–286.

[12] N. Williams, S. Zander, and G. J. Armitage, "A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification." Computer Communication Review, pp. 5–16, 2006.

[13] A. Este, F. Gringoli, and L. Salgarelli, "Support vector machines for tcp traffic classification," Computer Networks, pp. 2476–2490, 2009.

[14] M. Roughan, S. Sen, O. Spatscheck, and N. G. Duffield, "Class-of-service mapping for qos: a statistical signature-based approach to ip traffic classification." in Internet Measurement Conference, 2004, pp. 135–148.

[15] Mininet. http://mininet.org/

[16] D. Z. Moore, Andrew and M. Crogan, "Discriminators for use in flowbased classification." Queen Mary and Westfield College, Department of Computer Science, 2005.

[17] Luchaup, Daniel, et al. "Deep packet inspection with DFA-trees and parametrized language overapproximation." INFOCOM, 2014 Proceedings IEEE. IEEE, 2014.

[18] wireshark. https://www.wireshark.org/

[19] Tcp statistic and analysis tool, http://tstat.tlc.polito.it/index.shtml

[20] M. A. Hall, "Correlation-based feature selection for discrete and numeric class machine learning," in Proceedings of the 17th International Conference on Machine Learning, 2000, pp. 359–366.

[21] Lomax, Susan, and Sunil Vadera. "A survey of cost-sensitive decision tree induction algorithms." ACM Computing Surveys (CSUR) 45.2 (2013): 16.

[22] Weka. http://www.cs.waikato.ac.nz/ml/weka

[23] Mawi working group traffic archive. http://mawi.wide.ad.jp/mawi