Comparison of Internet Traffic Classification Tools

Hyunchul Kim, Marina Fomenkov, kc claffy, Nevil Brownlee
CAIDA, University of California San Diego
{hkim, marina, nevil, kc}@caida.org

Dhiman Barman, Michalis Faloutsos
University of California Riverside
{dhiman, michalis}@caida.org

I. Introduction

What is the best traffic classification method to date? Under what conditions? Why? Despite a plethora of research devoted to traffic classification and a variety of proposed traffic classification methods, the research community still does not have definitive answers to these questions, and the task of traffic classification remains unapproachable and confusing for a practitioner.

Rigorous comparison of various classification methods is challenging for three reasons. First, there is no publicly available payload trace set, so every method is evaluated using a different set of locally collected payload traces. Second, existing approaches use different techniques that track different features, tune different parameters and use different definitions and categorization of applications. Third, more often than not, authors do not make their developed implementation codes publicly available once they publish their results.

To address these challenges, we have conducted a comprehensive and coherent evaluation of three traffic classification approaches: port-based, behavior-based, and statistical. For each approach we selected a representative tool to test: CoralReef [1], BLINC [4], and WEKA [2], correspondingly. In this paper we present the results of our comparison, debunk traffic classification myths, identify caveats, and suggest practical tips.

II. Comparison methodology

A. Data set

The dataset we used for testing consisted of seven payload traces collected at two backbone and two edge links located in US, Japan, and Korea (Table I). The PAIX backbone traces were taken on OC48 links of an US Commercial Tier 1 backbone link that connects San Jose and Seattle. The WIDE trace was captured at a 100 Mb/s Ethernet US-Japan transoceanic link that carries commodity traffic for WIDE member organizations. The Keio traces were collected on a 1 Gb/s Ethernet link at Keio University Shonan-Fujisawa campus. The KAIST traces were captured at one of four external links connecting a 1 Gb/s KAIST campus network to KOREN, a national research network in Korea.

Diverse geographic locations, throughput, and application mix represented in these data allow us to test the traffic classification tools under a wide variety of conditions.

B. Metrics

To measure the performance of a classification method we use four metrics: *precision*, *recall*, *aggregate precision*, and *F-Measure* [7]. The *precision* of an algorithm is the ratio of True Positives over the sum of True Positives and False Positives[1], or the percentage of flows that are properly attributed to a given application by this algorithm. *Recall* is the ratio of True Positives over the sum of True Positives and False Negatives, or the percentage of flows in an application class that are correctly identified. *Aggregate precision* is the ratio of the sum of all True Positives to the sum of all the True Positives and False Positives for all classes. We apply the two former metrics to evaluate the quality of classification results for each application class and the latter metric to characterize the overall accuracy of a classifier on the whole trace set. Finally, *F-Measure* combines precision and recall into a single metric by taking their harmonic mean: $2 \times precision \times recall/(precision + recall)$. We use this metric to compare and rank per-application performance of 20 machine learning algorithms included in WEKA.

To establish a ground truth, we used the payload-based classifier [4] augmented with more signatures from [3] and manual payload inspection.

III. Results

A. CoralReef

Despite the common parlance that ports are no longer useful in identifying application, port-based tools such as CoralReef still achieve high precision and recall ($>$ 90%) for several legacy applications and protocols such as DNS, SNMP, NTP, News, Mail, Chat, SSH, and WWW. Port-based tools can classify these applications accurately because i) they mostly use default ports; and ii) their default ports are seldom used by other applications.

CoralReef fails to yield accurate classification results in the following three cases: (i) when applications mostly use ephemeral ports, e.g., P2P and passive FTP data transfer; (ii) when default ports of an application coincide with port masquerading P2P applications, e.g., streaming and game ports; and (iii) when default ports of an application overlap with those of others, e.g., SHOUTCAST Streaming uses port 8000, which is also used by some WWW servers.

B. BLINC

BLINC implements a behavior-based approach to traffic classification: it captures the profile of a host, in terms of destinations and ports the host talks to, identifies applications the host is engaged in, and then classifies traffic flows. For this

[1] True Positives is the number of correctly classified flows, False Positives is the number of flows falsely ascribed to a given application, and False Negatives is the number of flows from a given application that are falsely labeled as another application.

TABLE I. CHARACTERISTICS OF ANALYZED TRACES

Set	Date	Day	Start	Duration	Link type	Src.IP	Dst.IP	Packets	Bytes	Avg. Util	Avg. Flows (per 5 min.)	Payload
PAIX-I	2004-02-25	Wed	11:00	2h	backbone	410 K	7465 K	250 M	91 G	104 Mb/s	1055 K	16 Bytes
PAIX-II	2004-04-21	Wed	19:59	2h 2m	backbone	2275 K	17748 K	1529 M	891 G	997 Mb/s	4651 K	16 Bytes
WIDE	2006-03-03	Fri	22:45	55m	backbone	263 K	794 K	32 M	14 G	35 Mb/s	312 K	40 Bytes
Keio-I	2006-08-06	Tue	19:43	30m	edge	73 K	310 K	27 M	16 G	75 Mb/s	158 K	40 Bytes
Keio-II	2006-08-10	Thu	01:18	30m	edge	54 K	110 K	25 M	16 G	75 Mb/s	92 K	40 Bytes
KAIST-I	2006-09-10	Sun	02:52	48h 12m	edge	148 K	227 K	711 M	506 G	24 Mb/s	19 K	40 Bytes
KAIST-II	2006-09-14	Thu	16:37	21h 16m	edge	86 K	101 K	357 M	259 G	28 Mb/s	21 K	40 Bytes

study, we extended BLINC code to generate node profiles of not only source ⟨IP, port⟩ pairs but also of destination ⟨IP, port⟩ pairs. The modified code, Reverse BLINC, improved the aggregate precision on backbone traces by as much as 45%, since in those traces one of the two directions of traffic is often missing due to asymmetric routing. However, the code extension almost doubles the memory usage and running time.

BLINC has 28 different parameters to tune. For traces captured on the same link, the optimal threshold values remained nearly the same. For traces from different links, separate tuning was necessary to prevent degradation of the aggregate precision by 10%-20%. Our experience with BLINC classifier also suggests that one should tune parameters for P2P applications first, because almost every module in the code checks them.

Once tuned, BLINC classifies WWW, DNS, Mail, Chat, FTP, and Streaming flows with >90% precision. However, recall values for these applications are lower than precision values, since all classification is threshold-based: the number of application flows from a given source should exceed a certain threshold in order to trigger a classification attempt. If there are not enough flows from this source, then this traffic remains unclassified. DNS, Mail, and Chat have lower recall in backbone traces than in edge traces, because even Reverse BLINC could not capture those application flows when server flows were missing from backbone traces. Recall on FTP, Streaming, and Game is always lower than 25.8% across all traces, since behavior signatures of BLINC for these applications do not cover the following cases: (i) when a Streaming or FTP server concurrently provides any other application services; (ii) when a Game client sends any TCP flows or talks to only a small number of destination hosts.

With proper tuning, BLINC reliably identifies P2P flows, particularly when we apply CoralReef first to filter out DNS flows which BLINC often misclassifies as P2P. When applied to the remaining flows, BLINC achieves >85% precision in terms of flows on P2P applications. However, recall of P2P traffic in terms of bytes is significantly lower than flow recall. We conjecture that this difference in recall is due to the fact that some P2P applications usually assign different ephemeral ports for every single data transfer. If such transfers are large, then they account for a large number of bytes, but the number of flows remains below the classification triggering threshold, and, therefore, this traffic remains unclassified.

C. Machine Learning Algorithms

We have evaluated 20 different machine learning algorithms from the WEKA library using their default parameters. To select the most discriminating features from the 34 possible attributes, we tried both consistency-based feature selection and correlation-based feature selection with the BestFirst search method. For different traces, 5-10 features were selected for discriminating. Correlation filters produced better feature sets leading to a higher aggregate precision than consistency filters did, which is consistent with the results of [5], [6].

We then divided each trace into a set of 5 minute interval subtraces and trained algorithms on each sub-trace. Next, we tested the resulting classifier on several other sub-traces, by averaging the results over all runs.

The Random Forest (Decision Tree based) algorithm [7] showed the best aggregate precision (>80% for every trace) not only among the 20 different machine learning algorithms but among all the techniques that we have evaluated in this study. However, its per-application performance highly varies across different applications and traces.

Generally, Rule-based and Decision Tree-based algorithms [7] outperformed others in both aggregate precision and per-application F-Measure. However, we could not find any single machine learning algorithm that would yield >90% precision and recall on all applications even in a single trace. We are still experimenting with tuning parameters of machine learning algorithms including training algorithms using different training sets with different sampling size, application mix, and feature sets to investigate their impacts on aggregate precision, per-application precision and recall. We are also finding out key features for each application in all of our traces.

IV. CONCLUSIONS

For every method we evaluated, P2P, Games, and Streaming applications were harder to identify than other conventional applications. Unique characteristics of these applications may not be captured by currently available techniques which focus on only one type of information: port number, behavior pattern, or flow characteristics. We propose to build a combined classifier where existing techniques are carefully combined based on their per-application performance. We are currently experimenting with various combinations of CoralReef, BLINC, and machine learning algorithms. Preliminary results look promising. We intend to present our future findings at the Workshop.

REFERENCES

[1] *CoralReef*. http://www.caida.org/tools/measurement/coralreef/.

[2] *WEKA: Data Mining Software in Java*. http://www.cs.waikato.ac.nz/ml/weka/.

[3] J. Erman, M. Arlitt, and A. Mahanti. Traffic Classification Using Clustering Algorithms. In *ACM SIGCOMM MineNet Workshop*, 2006.

[4] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. Blinc: Multilevel traffic classification in the dark. In *ACM SIGCOMM*, 2005.

[5] A. Moore and D. Zuev. Internet traffic classification using Bayesian analysis techniques. In *ACM SIGMETRICS*, 2005.

[6] N. Williams, S. Zander, and G. Armitage. A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification. *ACM SIGCOMM CCR*, October 2006.

[7] I. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. Morgan Kaufmann, 2005.