Traffic Classification on Backbone Using Compact Protocol Fingerprints

Jin Zhang Qinrang Liu Xiaona Niu
National Digital Switching System Engineering and Technology Research Center (NDSC)
Jianxue street 7#, Box. 1001, No.783, 450002, Zhengzhou, China
{zhangjin, liuqinrang, niuxiaona}@mail.ndsc.com.cn

***Abstract*—Traffic classification using statistical characteristics (or fingerprints) of IP flows such as packet size and packet inter-arrival time has showed its preliminary success in sense of accuracy and simplicity. However, the large size of the fingerprints makes it impractical to deploy such method on backbone networks where a both large and high-speed memory is needed to accomadate the fingerprints and to catch up with the high packet rate. To overcome this drawback, in this paper we proposed a new idea and method that classifying network traffic using *compact* protocol fingerprints instead of using the naïve ones. We presented Two-Stage Compacting (TSC) algorithm to compress the protocol fingerprints. The TSC algorithm firstly performs distributional quantification on each feature dimension and then merges the feature vectors which incur similar inter-class distributions into a cluster, thus reduce the size of fingerprints remarkably. We demonstrate the effectiveness of our approach on real traffic traces. The experimental results show that a compression ratio of 32:1 can be achieved with nominal accuracy loss and reasonable computation overhead increase. Our results indicate that it is feasible to perform real-time traffic classification on high-speed backbone networks with current technical constraints.**

Keywords- traffic classification, statistical characters, compression

I. INTRODUCTION

The past few years have witnessed a dramatic increase in the number and variety of applications running over the Internet and over enterprise IP networks. Classifying network traffic according to the applications that generate them is fundamental to many traffic management tasks such as QoS level mapping, application specific traffic engineering, capacity planning, provisioning and security.

Traffic classification according to port numbers is a very common method that has been accepted by both the industrial and research community. However, recent research [3-16] reveals that this approach is not effective any more because more and more applications (such as P2P file sharing or VoIP) use non-standard port numbers or protocol tunneling techniques to avoid being detected. Instead of mapping traffic into application class according to port numbers, newly proposed methods rely on packet payloads [1-4], communication patterns [5, 6] or statistical traffic characters [7-16] to classify traffic. Although these newly proposed methods are more accurate than port number based one, most of them [1-14] are computationally ineffective, thus are inappropriate for real-time traffic classification on high-speed backbone networks.

A very recent work of Crotti et al. [15, 16] introduced a simple and accuracy classification method that relay on the joint distribution of packet size and inter-arrival of the first several packets (they call it "protocol fingerprints") in an IP flow. Although at its very early stage of development, this method can classify several types of protocols with high accuracy. The main drawback of such method is that it relays on large size (over 1Gbits) protocol fingerprints. When deployed on backbone networks, it must use high speed memories (such as SRAM) to keep up with the high packet rate. However, a large high-speed memory is always expensive.

Inspired by the previous work of Crotti et al. [15, 16], we presented a new traffic classification method that relay on compact protocol fingerprints. We deployed the protocol fingerprints building-up process of [15, 16]. Furthermore, we proposed a Two-Stage Compacting (TSC) algorithm to compress the naïve protocol fingerprints into a compact one. The virtue of TSC is that it takes advantage of non-uniform distribution on each feature dimension thus incurs a much lower overhead than existing compression algorithm[1], such as Distributional Clustering (DC) [18]. For the sake of comparison, we evaluated classification accuracy using both the naïve protocol fingerprints and the compressed one under various compression ratios. Our experiments show that a compression ratio of 32:1 can be achieved with merely 4% loss in classification accuracy. This means real-time traffic classification on backbone networks is feasible using a compact protocol fingerprints that can be accommodated in a fast, small memory.

The rest of the paper is organized as follows. The definition of protocol fingerprints and the classification algorithm is briefly introduced in section 2. In section 3 we describe the TSC algorithm and discuss the computational overhead introduced by using compact protocol fingerprint. Experimental results are given in section 4. We conclude this paper and point out the future direction of our work in section 5.

II. CLASSIFICATION USING NAÏVE PROTOCOL FINGERPRINTS

Our objective of traffic classification is to map a unidirectional IP flow into one of the total M protocol classes

[1] See section 3.C for detailed analysis.

C_j ($1 \le j \le M$). We define flow F as the unidirectional, ordered sequence of IP packets produced either by the client towards the server, or by the server towards the client during an application layer session [15, 16]. Assume the first N sequential packets of F are PKT_1 , PKT_2 ,…, PKT_N . The *i*-th packet of F , PKT_i , is characterized by a two-dimension **character pair** $P_i = (s_i, t_i)$ ($1 \le i \le N$) where s_i and t_i ($t_1 = 0$) are packet size of PKT_i and packet inter-arrival time between PKT_i and PKT_{i-1} respectively. We call $\vec{P} = (P_1, P_2, ..., P_N)$ **character vector** of F . Given enough pre-classified training flow F of class C_j , P_i forms a tow-dimension probability table PDF_i^j , and the probability table vector $P\vec{D}F^j = (PDF_1^j, PDF_2^j, ..., PDF_N^j)$ is defined as "**protocol fingerprint**", which represents the statistical character of flows from class C_j . The *i*-th "**fingerprints slice**" of all the protocols is defined as $P\vec{D}F_i = (PDF_i^1, PDF_i^2, ..., PDF_i^M)$. The rational behind traffic classification using character vector $\vec{P}$ of a flow is that $\vec{P}$ captures the application's negotiation phase, which is usually a pre-defined sequence of messages different between applications [13, 14].

After building up the protocol fingerprints of each class, a flow F (assume its character pair is $\vec{P}$) is assigned to the class whose protocol fingerprint is most statistically similar to $\vec{P}$. A statistical metric called *normalized anomaly score* [15] is defined to represent the similarity between $\vec{P}$ and $P\vec{D}F^j$. A flow is assigned to the protocol class where it gets a minimum normalized anomaly score. The detailed description of the classification algorithm can be found in [15, 16]. *This classification method obtains remarkable computational simplicity at the cost of notable spatial complexity.*

III. COMPACT PROTOCOL FINGERPRINTS

The notable size of naïve protocol fingerprint is demonstrated below. Assume the number of possible packet length is *L* and the number of possible packet inter-arrival time is *T*. Then the size of total fingerprints is $M \cdot N \cdot L \cdot T$. We choose the range of packet size from 40 to 1500 (bytes) and that of packet inter-arrival time t as 10^{-7} to 10^{3} (seconds) as the previous work [15] do. Considering the large magnitude t ranging in, we use $\log_{10}(t)$ with a step 0.01 instead [15]. Assume $M = N = 5$ and the probability values are 4-byte float or integer number, thus the total protocol fingerprints take 5*5*1461*1001>1Gbits memory, which is far beyond the capacity of today's high speed memories such as SRAM.

As a result, a compact protocol fingerprint that is both being small and having little negative impact on classification accuracy is needed for real-time high-speed traffic classification. In this section we introduce the Two-Stage Compacting (TSC) algorithms to build compact protocol fingerprints. Furthermore, we also discussed the overhead induced by using compact protocol fingerprints.

A. *Stage 1 : Distributional Quantification*

A naïve method to compress the probability density table *PDF* (in this section we omit subscript *i* for the sake of conciseness) is Straightforward Quantification (SQ), that is, binning packet length dimension (*L*-dimension) and packet inter-arrival time dimension (*T*-dimension) every δ_1^L and δ_1^T consecutive value into a group respectively, where $\delta_1 = \delta_1^L \cdot \delta_1^T$ is compression ratio of DQ. However, the extensive research work done in network measurement area reveal that traffic character distributions such as packet size and inter-arrival time are far from uniform. As a result, the straightforward quantification method will bias *PDF* highly. We introduce a simple algorithm called Distributional Quantification (DQ) to compress *PDF*. *The key design philosophy of DQ is to make the quantification resolution fine on densely distributed regions and coarse on sparsely distributed regions*. Intuitively, the compact protocol fingerprints generated by DQ will be more distinguishable than those generated by SQ.

Assume the probability density function and the cumulative distribution function of discrete random variable X is $f(x)$ and $F(x)$ respectively, both of which are defined on discrete set $\{1, 2, ..., n\}$. Let $P(a,b) = F(b) - F(a)$, where $a, b \in [1, n]$ and $a \le b$. Assume the set of quantification low bounds is $\vec{l} = (l_1, l_2, ..., l_m)$ ($l_1 = 1$), $m = \lfloor n/\delta \rfloor$ where δ is the compression ratio. The detail of DQ is described in figure 1. $length(\vec{l})$ And $upper(\vec{l})$ means the number of elements in set $\vec{l}$ and the top-most element of $\vec{l}$ respectively. Because we can not know the actual probability distribution of packet size and inter-arrival time, we use the empirical distribution derived from traffic statistics instead. We apply the DQ algorithm on *L*-dimension and *T*-dimension of the probability density table *PDF* independently. The resulting compressed probability density table is *PDF'* .

1: $\vec{l} \leftarrow \{1\}$, $m \leftarrow \lfloor n/\delta \rfloor$, *total* = 1, *i* = 1;
2: while ($i \le n$) and ($length(\vec{l}) < m$)
3: if $P(upper(\vec{l}), i) \ge \frac{total}{m - length(\vec{l}) + 1}$
4: $\vec{l} \leftarrow \vec{l} \cup \{i\}$;
5: $total \leftarrow total - P(upper(\vec{l}), i)$;
6: end if;
7: $i \leftarrow i + 1$;
8: end while;

Figure 1. Distributional Quantification Algorithm

B. *Stage 2 : Distributional Clustering*

Distributional Clustering (DC) [18] is an effective algorithm proposed by pattern recognition community for the purpose of large probability table compression. We apply DC to each fingerprints slice $P\vec{D}F_i'$ produced by DQ into a

compact one, which has a large reduction in size and can be accommodated by a small size memory. To our knowledge, this paper is the first work that applying this technique to and evaluation its effect on network traffic classification.

The idea of DC is clustering packet character pairs that induce similar class probability distribution. Consider the random variable over protocol class, c, and its distribution given a particular packet character pair p_i, denoted $P(c \mid p_i)$. If there exists two packet character pair p_i and p_j which induce similar class distributions, then they can be clustered together and represented by a single distribution $P(c \mid p_i \vee p_j)$, where

$$P(c \mid p_i \vee p_j) = \frac{P(p_i)P(c \mid p_i) + P(p_j)P(c \mid p_j)}{P(p_i) + P(p_j)} \tag{1}$$

The "weighted mean KL divergence" is applied as the metrics of similarity between two distributions, that is

$$\begin{aligned} D_f(p_i, p_j) = P(p_i) \cdot D(P(c \mid p_i) \| P(c \mid p_i \vee p_j)) + \\ P(p_j) \cdot D(P(c \mid p_j) \| P(c \mid p_i \vee p_j)) \end{aligned} \tag{2}$$

Where

$$D(P(c \mid p_i) \| P(c \mid p_j)) = -\sum_{k=1}^{M} p(c_k \mid p_i) \log\left(\frac{p(c_k \mid p_i)}{p(c_k \mid p_j)}\right) \tag{3}$$

is the Kullback-Leibler (KL) divergence.

We adopted the two-stage clustering method proposed by Jianying Hu et al [18] to generate the compressed probability table. For a detailed description of the clustering algorithm please refer to [18]. Assume the compression ration of DC is δ_2, thus the compression ration of TSC is $\delta = \delta_1 \cdot \delta_2$. The whole TSC algorithm is described in figure 2.

Algorithm : TSC
Input : $PDF_i^j (1 \le i \le N, 1 \le j \le M)$
Output : $PDF_i'' (1 \le i \le N)$
1: for all the i and j :
2: $PDF_i^{j\prime} = DQ(PDF_i^j)$;
3: for i = 1 : N
4: $PDF_i'' = DC(PDF_i')$;
5: end

Figure 2. Figure 2 Two-Stage Compacting (TSC) Algorithm

C. Overhead Analysis

Using compact protocol fingerprints induce two kinds of overheads into the classification procedure. The first kind of overhead is the space complexity of storing mapping tables, while the second one is the time complexity of the mapping procedure.

A difference between our algorithm and the previous ones [15, 16] is that when accessing the probability density table, our algorithm takes an extra lookup in the map table in order to transform the table index from *PDF* to *PDF″*. The map table size S_{MT} is :

$$S_{MT} = N \cdot M \cdot L \cdot T \cdot \log_2(M \cdot L \cdot T + \frac{1}{\delta} \cdot M \cdot L \cdot T) \tag{4}$$

The extra memory access time is 1 per packet. In fact, there is a tradeoff between map table size and time complexity of map table looking-up. If we build map table on *L*-dimension and *T*-dimension separately, then the map table size S_{MT}' is :

$$\begin{aligned} S_{MT}' = N \cdot M \cdot (L \cdot \log_2(M \cdot L + \frac{1}{\delta_1^L} \cdot M \cdot L) + \\ T \cdot \log_2(M \cdot T + \frac{1}{\delta_1^T} \cdot M \cdot T) + \\ \frac{L \cdot T}{\delta_1} \cdot \log_2(\frac{M \cdot L \cdot T}{\delta_1} + \frac{M \cdot L \cdot T}{\delta})) \end{aligned} \tag{5}$$

Obviously $S_{MT}' < S_{MT}$. However, the extra memory access time is 2 per packet in this case.

IV. Experimental Results

A. Testbed Setup

We evaluated the accuracy of classification using compact protocol fingerprints with real world traffic traces collected on gateway switch of our campus residential network. The campus residential network connects about 1000 home or dormitory PCs to the Internet through a 1 Gbps Ethernet link. We collected four days of working day traffic traces from Monday to Thursday, on each day we choose three typical periods, that is 9:00-11:00, 15:00-17:00, and 20:00-22:00. The total data size is about 230GBytes, with a mean link bandwidth of 21.3Mbps. We use traffic traces collected from Monday to Tuesday as the training data and those collected from Wednesday to Thursday as the evaluation data.

In this paper we use only four classes of traffic, that is: HTTP, POP3, SMTP and BitTorrent[2], which compose more than 85% of the total traffic, and test if the algorithm can effectively map an unknown flow to the right class. We build base truth for both training data and evaluation data using the payload-based approach described in [6]. The protocol fingerprints are built up using the first 5 packets of each flow belonging to certain classes. Previous work shows classification using IP flows send by the client to server is more accuracy than using those send from server to the client [15, 16], so we only consider the previous and ignore the latter in this paper.

B. Comparison with Naïve Protocol Fingerprint

We evaluated the accuracy of classification using both the naïve protocol fingerprints and the compact one under two

[2] Just as [15, 16], we focus on very limited application types in this paper. The extension of our work to other protocols is left as future work. We only want to test the feasibility of compact protocol fingerprints in this paper.

metrics: *hit ratio* and *false positive ratio*. The definitions of these two metrics are given below.

Definition *Hit Ratio* and *False Positive Ratio*. Assume the set of flows of protocol type p in evaluation data set is E^p, the set of flows that are regarded as protocol type p is $\hat{E}^p$, then the *Hit Ratio* and *False Positive Ratio* are define as (6.a) and (6.b) respectively.

$$HR = \frac{|E^p \bigcap \hat{E}^p|}{|E^p|} \quad (a) \qquad FPR = \frac{|\hat{E}^p - (E^p \bigcap \hat{E}^p)|}{|\hat{E}^p|} \quad (b) \quad (6)$$

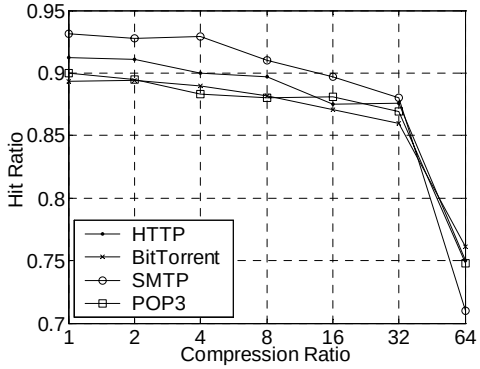

Figure 3 Hit Ratio under various compression ratio

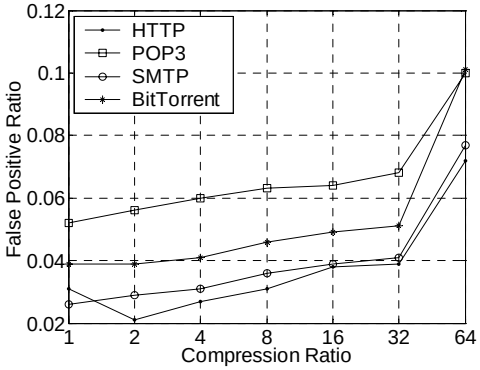

Figure 4 False Positive Ratio under various compression ratio

Figure 3 and Figure 4 are *hit ratio* and *false positive ratio* under various compression ratios. Note that the accuracy of using naïve protocol fingerprints is not printed explicitly on the figures because it is right the case when compression ratio equals to 1.

From figure 3 and figure 4 we can see that a compression ratio of 32:1 is feasible with no more than 4% loss in hit ratios and around 1.3% rise in false positive ratios. This means it is feasible to classify traffic using compact protocol fingerprints.

V. CONCLUSIONS AND FUTURE WORK

Aiming at building a real-time traffic classification system on high-speed backbone networks, we proposed an accuracy and efficient classification method based on compact protocol fingerprints. Compared to the naïve protocol fingerprints, the compact ones are times of orders small and cause a acceptable performance degrade. Thus the method of traffic classification using compact protocol fingerprints is very appropriate for building a real-time traffic classification system on high-speed backbone networks. As an ongoing work, we are evaluating other compression methods that are developed by the computational pattern recognition community [17]. We are also engaged in building a payload-based, pattern-matching traffic classification tools that can classify the majority of our traffic traces (say, more than 95%) with high accuracy and apply our method on classifying more traffic classes such as P2P file-sharing and VoIP traffic.

REFERENCES

[1] V.Paxson. Bro: a system for detecting network intruders in real-time. Computer Networks, 31(23-24):2435-2463, 1999.

[2] M.Roesch. SNORT: Lightweight Intrusion Detection for Networks. In LISA'99: Proceedings of the 13th USENIX Conference on Systems Administration, pages 229-238, Seattle, WA, USA, November 1999.

[3] P. Haffner, S. Sen, O. Spatscheck, and D. Wang. ACAS: Automated Construction of Application Signatures. In SIGCOMM'05 MineNet Workshop, Philadelphia, USA, August 2005.

[4] J. Ma, K. Levchenko, C. Krebich, S. Savage, and G. Voelker. Unexpected Means of Protocol Inference. In IMC'06, Rio de Janeiro, Brasil, October 2006.

[5] T. Karagiannis, A.Broido, M. Faloutsos, and kc claffy. Transport layer identification of P2P traffic. In ACM/SIGCOMM IMC, 2004.

[6] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: multilevel traffic classification in the dark. In SIGCOMM'05 pages 229–240, Philadelphia, PA, USA, August 2005.

[7] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield. Class-of-Service Mapping for QoS: A Statistical Signature-based Approach to IP Traffic Classification. In IMC'04, Taormina, Italy, October 2004.

[8] A. McGregor, M. Hall, P. Lorier, and J. Brunskill. Flow Clustering Using Machine Learning Techniques. In PAM 2004, Antibes Juan-les-Pins, France, April 2004.

[9] A. W. Moore and D. Zuev. Internet traffic classification using bayesian analysis techniques. In SIGMETRICS'05, pages 50–60, Banff, Alberta, Canada, June 2005.

[10] Jeffrey Erman, Anirban Mahanti, and Martin Arlitt. Internet Traffic Identification using Machine Learning. In GlobeCom 2006.

[11] J. Erman, M. Arlitt, and A. Mahanti. Traffic Classification using Clustering Algorithms. In SIGCOMM'06 MineNet Workshop, Pisa, Italy, September 2006.

[12] J. Erman, M. Arlitt, A. Mahanti, and C. Williamson. Identifying and Discriminating Between Web and Peer-to-Peer Traffic in the Network Core. In WWW'07, Banff, Canada, May 2007.

[13] Laurent Bernaille, Renata Teixeira, Ismael Akodkenou, et al. Traffic Classification On The Fly. ACM SIGCOMM Computer Communication Review. Volume 36, Number 2, 23-26, April 2006.

[14] L. Bernaille, R. Teixeira, and K. Salamatian. Early Application Identification. In The 2nd ADETTI/ISCTE CoNEXT Conference, Lisboa, Portugal, December 2006.

[15] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli. Traffic Classification through Simple Statistical Fingerprinting. Computer Communications Review, 37(1):7-16, 2007.

[16] Manuel Crotti, Maurizio Dusi, Francesco Gringoli, Luca Salgarelli. detecting http tunnels with statistical mechanisums. In ICC 2007. Glasgow, Scotland, June 2007.

[17] Sung-Jung Cho, Michael Perrone and Eugene Ratzlaff. EM mixture model probability table compression. In IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP). Page: II-813-16, vol.2. 2003.

[18] Jianying Hu and Eugene Ratzlaff. Probability Table Compression Using Distributional Clustering for Scanning N-Tuple Classifiers. In 17th International Conference on Pattern Recognition (ICPR). Vol.2, pp. 533-536. 2004.