Traffic Classification Based on Flow Similarity

Jae Yoon Chung[1], Byungchul Park[1], Young J. Won[1], John Strassner[1,2], and James W. Hong[1]

[1] Dept. of Computer Science and Engineering, POSTECH, Korea
[2] Waterford Institute of Technology, Waterford, Ireland
{dejavu94,fates,yjwon,johns,jwkhong}@postech.ac.kr

Abstract. Due to the various masquerading strategies adopted by newer P2P applications to avoid detection and filtering, well-known port mapping techniques cannot guarantee their accuracy any more. Alternative approaches, application-signature mapping, behavior-based analysis, and machine learning based classification methods, show more promising accuracy. However, these methods still have complexity issues. This paper provides a new classification method which utilizes cosine similarity between network flows.

Keywords: Traffic classification, traffic monitoring.

1 Introduction

The large amount and diversity of today's Internet traffic requires more efficient and accurate traffic classification methods. For example, many modern network applications, such as Voice over IP (VoIP), use features such as dynamic port allocation, [*]incorporate ephemeral port allocation (e.g. peer to peer (P2P) and web storage service applications); some applications also allocate their traffic to well-known ports to avoid detection (e.g. P2P and YouTube). Therefore, port-based classification [1] cannot guarantee that it can correctly identify applications by inspecting the ports used [5]. To overcome this issue, alternative traffic classification techniques, such as signature-based classification [4][7][8][13], behavior-based modeling [3][6][12][18], and machine learning-based classification [9][10][11], are introduced.

In this paper, we propose a new traffic classification methodology that translates packet payloads into vectors and classifies the application traffic based on the result of similarity comparisons among the series of vectors. Our work is inspired by document classification [15], which is widely used in the Natural Language Processing (NLP). The basic idea of comparing packet payload vectors is analogous to document classification; however, there are fundamental differences between document content and packet payloads. Whereas a document consists of basic components called words which have precise meaning, packet payloads consist of binary data, of which most has no pre-defined meaning. Therefore, we try to convert packet payloads into a series of vectors reflecting this difference.

[*] This work was partly supported by the IT R&D program of MKE/IITA [2008-F-016-02, CASFI] and WCU program through the KSEF of MEST, Korea [R31-2008-000-10100-0].

The remainder of this paper is organized as follows. Section 2 provides an overview of application traffic classification methodologies. Section 3 describes our classification methodology based on flow similarity. The evaluation results with real-network traffic traces are presented in Section 4. Finally, the concluding remarks and future work are discussed in Section 5.

2 Related Work

Many previous studies on application traffic identification have considered well-known network ports [1] as the primary means for identifying traffic. However, these traditional port mapping methods now have a much lower identification accuracy, due to three factors: (1) increased use of dynamic port allocation, (2) the increased use of ephemeral ports, and (3) the use of well-known ports (e.g., a P2P application may use port 80, even though it is not using HTTP) to avoid detection and filtering. Such phenomena have degraded the accuracy of port mapping mechanisms. Moore et al. [2] hypothesized that the accuracy of port mapping methods is not more than 50~70%. Our technique does not use port identification for these reasons.

An alternative approach, signature-based classification, has been proposed to address the drawbacks of port-based classification [4][7][8][13]. Packet signature-based classification inspects the packet payload to identify the application. Other types of signatures-based mechanisms construct an application signature based on communication patterns or other identifying features. In either case, a signature is static and can be used to distinguish applications from each other. Once a reliable signature is available, this approach can identify the target application very accurately. However, generating the signature requires exhaustive searching and time consuming.

Other research focused on behavior-based classification methods [3][6][12][18], which characterize the behavior of communicating nodes using connection related information, such as tuples that identify the particular entity (e.g., in terms of the source destination addresses as well as other metrics, such as protocol number). BLINC [3] profiles host behavior using destination IP addresses and port allocation patterns; application traffic is then identified by comparing it with existing profiles. Behavior-based classification is efficient for detecting anomalies or worm-like P2P traffic [6], and is available with encrypted packet payloads. These methods do not rely on any information present in the payload; thus, its accuracy is often questionable. In addition, the time complexity is problematic due to obscure behavior patterns when dealing with a high volume of traffic.

More recently, machine learning-based approaches [9][10][11] using statistical information of the transport layer have been introduced. These classification methods can use unsupervised, supervised, reinforcement, or hybrid learning mechanisms to automatically classify packets or even flows based on (for example) their statistical characteristics, instead of fixed data like port numbers. In other words, different types of applications can be distinguished by their communication characteristics such as connection duration, inter-packet arrival time, and packet size. Since most machine learning-based classification methods only use statistical information, these methods can also be used with encrypted traffic. However, machine learning approaches have their own set of issues that our approach does not have, such as effective traffic feature selection. We focus on how to utilize the payload data without heavy packet inspection while achieving reasonable accuracy.

3 Proposed Methodology

This section explains our traffic classification method. Its strength is that it does not rely on extracting any packet information (e.g., from the IP header). We illustrate our payload vector converting, vector comparison, and flow comparison methods.

3.1 Vector Space Modeling

Vector Space Modeling (VSM) is an algebraic model from the NLP research area that represents text documents as vectors, and is widely used for document classification. The goal of document classification is to find a subset of documents from a set of stored text documents D which satisfy certain information requests or queries Q. One of the simplest ways to do this is to determine the significance of a term that appears in the set of documents D [17]. This is quite similar to traffic classification, which identifies and classifies network traffic according to the type of application that generated the traffic. The main issue in VSM is how to determine the weight of each term. Salton et al. [15] have proposed some recommended combination of term-weighting components: (1) Term Frequency Component (b, t, n), (2) Collection Frequency Component (x, f, p), and (3) Normalization Component (x, c). However, these combinations are not suitable for traffic classification, because these were derived based on the needs of document classification. We therefore have to consider other weighting methods in order to use VSM for traffic classification; these methods should correspond to those portions of the traffic payload that are application-specific and disregard other data.

3.2 Similarity

Once packet payloads are translated into vectors, the similarity between packet payloads can be calculated by computing the similarity between vectors using many metrics. We used one such measure – the cosine similarity, which measures the similarity between two vectors of *n* dimensions by finding the cosine of the angle between them. The similarity value ranges from -1 (meaning exactly opposite) to 1 (exactly same) and 0 indicates independence. We normalized payload vectors so that the similarity can be calculated by the dot product, and it approaches one if the vectors are similar and zero otherwise. Formula (1) is a mathematical definition of cosine similarity and Figure 1 illustrates the cosine similarity between payload vectors.

Our payload vectors have very high dimensionality (for example, 65536), so probabilistically they are very sensitive (the detail of vector converting is described in the next section). If two payload vectors are generated by the different applications, the contents of each payload consist of distinct word (or binary) sequence and their vectors are also very different. Because most signatures of the application traffic are a small portion of the payload data, we may ignore the other part of the payloads signatures which is actually arbitrary binary data.

$$Similarity(p_1, p_2) = \frac{V(p_1) \cdot V(p_2)}{|V(p_1)||V(p_2)|}. \quad (1)$$

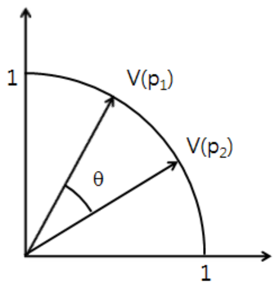


Fig. 1. Cosine similarity illustrated

3.3 Payload Vector

We define the *word* of a payload as follows. If a window size is longer, the payload vector can reflect the order of signature in the payload. However, we have to handle the high dimension vectors caused by increasing length of words.

- **Definition 1:** We define *word* as a payload data within an i-bytes sliding window where the position of the sliding window can be 1, 2 … n-i+1 with n-length payload. The size of whole representative words is 2^{8*i}.

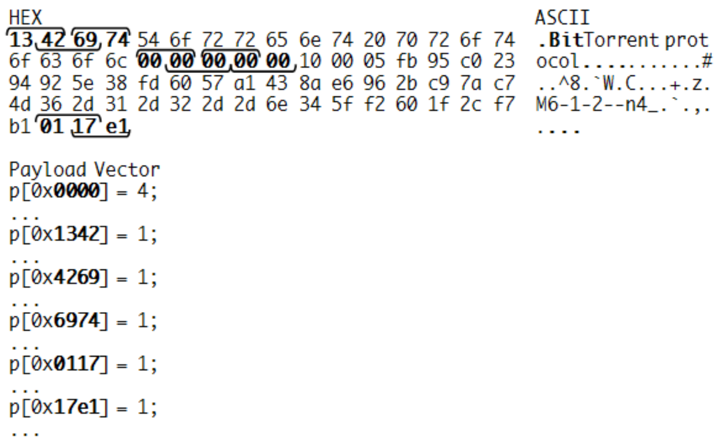
We can make the term-frequency vector by counting words within a payload. This vector is called the *payload vector*, which is the same as the document vector in NLP research.

- **Definition 2:** When w_i is the count of the i-th word that appears repeatedly in a payload, the *payload vector* is

$$Payload\ Vector = [w_1\, w_2\, ...\, w_n]^T. \quad (2)$$

where n is the size of whole representative words.

Our study focuses a window size of two, which means that the word size is two, because it is the simplest case for representing the order of content in payloads. When the word size is 2-bytes, the size of all representative words is 2^{16}=65536. Thus, the payload vector has 65536 dimensions. Figure 2 is part of a payload vector that is converted from a BitTorrent packet. In most of cases, we can see that the vector is very sparse because the dimension of the payload vector is much higher than the length of the payload.



```
HEX                                                ASCII
13 42 69 74 54 6f 72 72 65 6e 74 20 70 72 6f 74    .BitTorrent prot
6f 63 6f 6c 00 00 00 00 00 10 00 05 fb 95 c0 23    ocol..........#
94 92 5e 38 fd 60 57 a1 43 8a e6 96 2b c9 7a c7    ..^8.`W.C...+.z.
4d 36 2d 31 2d 32 2d 2d 6e 34 5f f2 60 1f 2c f7    M6-1-2--n4_.`.,.
b1 01 17 e1                                        ....

Payload Vector
p[0x0000] = 4;
...
p[0x1342] = 1;
...
p[0x4269] = 1;
...
p[0x6974] = 1;
...
p[0x0117] = 1;
...
p[0x17e1] = 1;
...
```

Fig. 2. Converting BitTorrent packet into payload vector

3.4 Packet Comparison

We have to calculate cosine similarities between payloads. In the document classification research, some studies enhance not only words that appear very frequently in a document, but also words that appear very rarely in the documents. When we calculate similarities, we have to define the term weight differently from the recommendations in [15]. However, if we apply the ‘term frequency – inverse document frequency’ weighting rule in our packet comparison process, the classification accuracy can be lower than that of typical signature-based classification approaches due to frequent appearances of signature. Therefore, we define weight values as the number of word appearances in a payload. These weight values are empirically updated.

If a packet is composed by w*, which appears in more than half of the payload length, then the similarity between a given packet and another packet that has at least one w* is always high. This is true even if those packets are generated by different applications. Conversely, it is the most important evidence for identifying a flow if a certain packet is composed of one word. To overcome mis-classification caused by improper weighting to a dominant word, we define the following exceptional case. We use the hard clustering approach; thus, the similarity between the packets containing the same w* is either zero or one.

- **Exception Case:** Let w* be the dominant word in a packet. If a packet is mostly filled with w*, the content of another packet in being comparison would also be mostly filled with w*.

3.5 Flow Comparison

In the scope of this paper, traffic classification relies on how to define and distinguish the similar flows. Flow here refers to bidirectional-flow containing both in and out packets from host.

3.5.1 Payload Flow Matrix

Formula (3) defines the *payload flow matrix (PFM)*. The i-th row of a PFM is the i-th packet of the flow. PFM is a $k \times n$ matrix, where k is the number of packets and n is the dimension of payload vectors. It is defined as follows.

- **Definition3:** *Payload flow matrix (PFM)* is

$$PFM = [\, \overrightarrow{p_1}\, \overrightarrow{p_2} \dots \overrightarrow{p_k}\,]^T \tag{3}$$

where $\overrightarrow{p_i}$ is *payload vector* as defined in Section 3.3.

3.5.2 Collected PFMs

Figure 3 illustrates a collection of m PFMs ($k \times n \times m$ matrix). Each layer represents a typical flow of each application. The collected PFMs are empirically accumulated according to formula (4), where α is a constant weight value. They are not only the basis for classifying application traffic but also for defining the term frequency weighting. For comparison aspects, we do not need a large amount of PFMs as a training set, but we need to know the typical flows of each application.

$$Collected\ PFMs = \alpha * new\ PFM + (1 - \alpha) * Collected\ PFMs \tag{4}$$

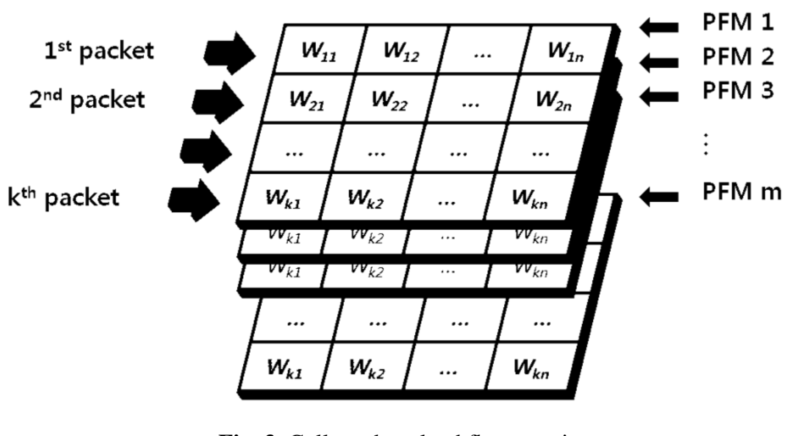


Fig. 3. Collected payload flow matrices

3.5.3 Flow Similarity Calculation

Figure 4 shows how to compare a new flow with two previously collected PFMs belonging to two different applications. The vertically linked circles refer to the packets in each flow. The dash line implies the comparison sequence between the packets. The first packet of the new flow is compared to only the first packets of each PFM. It means that we keep the order of packets within the flows. We assume that the two flows are well matched even if we keep the order of comparison sequences. Algorithm 1 is the pseudo-code representation of flow similarity calculation according to

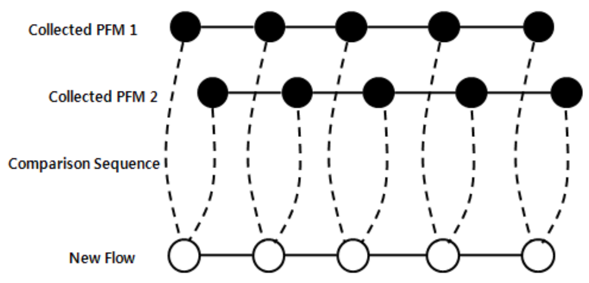


Fig. 4. Flow and packet comparisons

Algorithm 1. Pseudo code for flow similarity calculation

```
flow_similarity(collected_flows, input_flow)
begin
    N = the first N packets in the flow;
    n = the number of flows in the collected_flows;

    for i=1 to n,
        compared_flow = collected_flows(:,:,i);
        for j=1 to N
            weight = compared_flow(j,:)/row_sum(compared_flow(j,:));
            weighted_flow = input_flow(j,:).*weight;
            pkt_sim(j,i) = similarity(compared_flow(j,:), weighted_flow);
        end
    end

    /*
     * W_* is scoring weight
     * M is number of scoring packets
     */
    flow_similarity =
        pkt_sim(1, :)*W_1 + pkt_sim(2, :)*W_2 + ... + pkt_sim(M, :)*W_M;

    return flow_similarity;
end
```

Figure 4. We can determine the similarity scores for each collected PFM. If the score between the PFM 1 and new flow is higher than the score between the PFM 2 and new flow, we conclude that the new flow belongs to the application of PFM 1. However, when its similarity score does not exceed a certain threshold value, the new flow becomes unknown.

4 Evaluation

Our evaluation divides into two steps. The first step is to determine the best-fitting parameters for accurate traffic classification. Then, we evaluate the accuracy of our method using real network traffic from our campus backbone.

For flow similarity calculation, the following parameters should be determined: (1) weighting scheme, and (2) the number of first N packets in a flow. We select five target applications and compare the similarity scores for every pair of target applications with respect to various parameter settings. The selected target applications are BitTorrent [19], Emule [20], YouTube [21], Fileguri (P2P file-sharing application) [22], and Afreeca (P2P video streaming application) [23]. They are chosen based on their usage popularity in our campus network. In addition, YouTube and Afreeca are video streaming services, which enable us to check whether our method is valid for web-based video streaming detection.

We generated five flows of each application at an end host and *collected PFMs* (which were described in section 3.5.2). We captured another flow generated by one of target applications and calculated the similarity between this flow and five *collected*

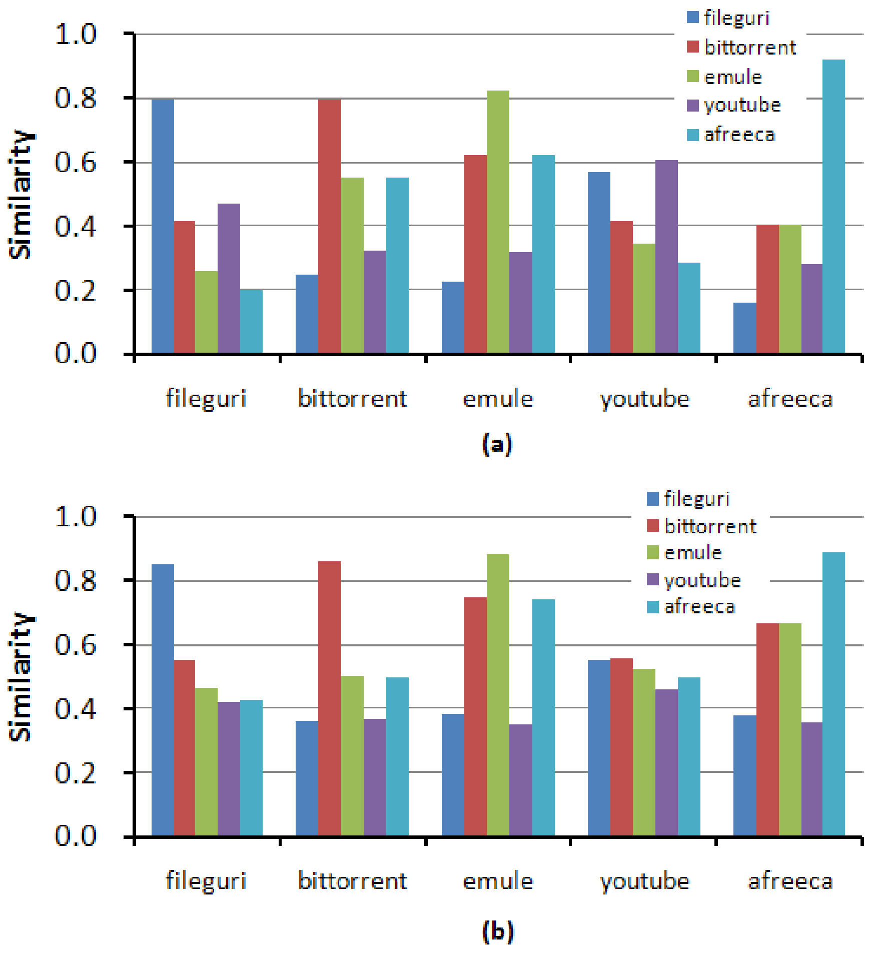
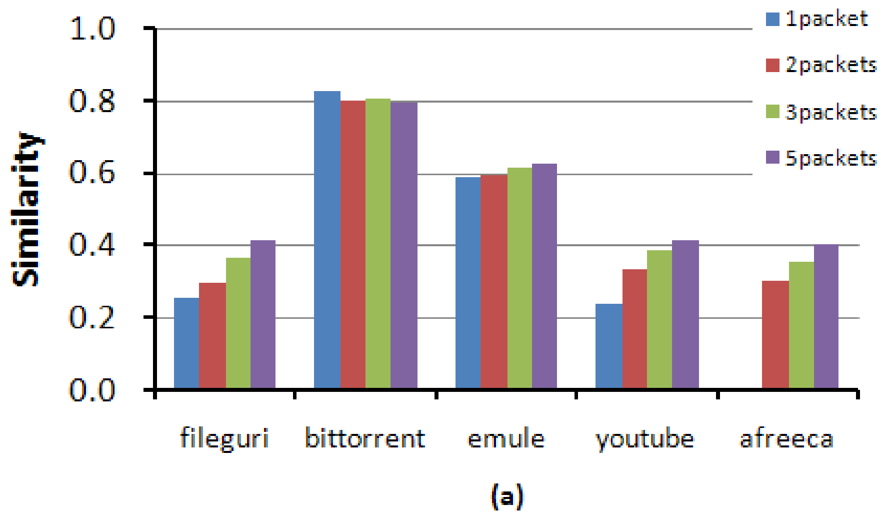


Fig. 5. Exponentially decrease weight (a) vs. uniform weight (b)

PFMs with two different weighting schemes; our two schemes are (a) exponentially decreasing weight and (b) uniform weight. Figure 5 shows the variance of similarities between application traffic is larger if the weights are exponentially decreased. In other words, the results demonstrate the fact that the first few packets have remarkable characteristics or signatures that distinguish each flow [14].

We calculated the similarities between the single flow of a target application and *collected PFMs* with various N. Figure 6 (a) shows that the BitTorrent flow and the *collected PFM* of BitTorrent is almost the same, although N is changed from 1 to 5.



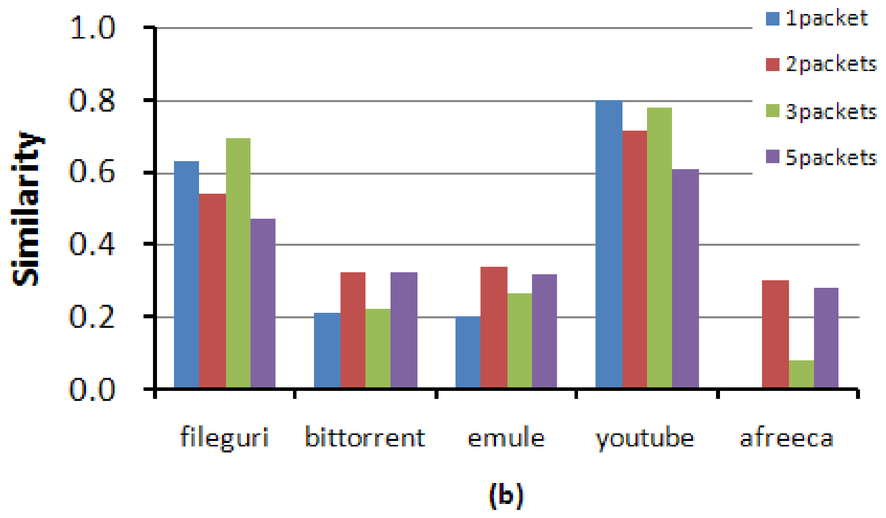


Fig. 6. Similarity with respect to changing first N (=1, 2, 3, 5) packets when BitTorrent (a) and YouTube (b) are the flows being compared to

Figure 6 (b) shows that YouTube flows are basically similar to Fileguri flows because both applications use the HTTP protocol to request download files or videos; this results in almost identical payload formats. Figure 6 (b) infers that a large N can help to distinguish flows that are generated by different applications having similar content. If N is too large, however, the later packets of a flow can contain arbitrary binary data. This can decrease the similarity, even when the flows were generated by the same application. From these observations, we chose exponentially decreasing weighting and N=2 (excluding data packets) as the best-fitting parameters for classifying selected target applications.

We validated our approach using the campus backbone traffic. The traces were collected from one of the two Internet junctions at POSTECH, a university with a user population of about 3500 people. To avoid any possible packet loss, the monitoring probe equipped with Endace DAG 4.3GE [24] was used to monitor 1 Gbps Ethernet link. The collected trace covers 40 minutes period (23:00 to 23:40) in early 2009. We took a sample of traffic from several end hosts and estimated the results within the whole backbone traffic.

The ground truth traffic is obtained by Traffic Measurement Agent (TMA). TMA is a Windows based client program and is deployed at target endpoints to determine the exact flow information at the origin of the traffic. TMA monitors the network interface of the host. It also summarizes traffic and records log data, including five flow tuples, process name, packet count, and time. The log data is compared with the results of classifications from POSTECH's Internet junction traffic; from this, we can calculate false positives and false negatives and remove them to compute the overall accuracy, as defined by formula (5). Table 1 explains the results of the classification of four applications: BitTorrent, LimeWire, YouTube, and Fileguri.

$$Overall\ Accuracy = \frac{Total\ traffic - (FP\ traffic + FN\ traffic)}{Total\ traffic} \quad (5)$$

BitTorrent has the most accurate results because it is well classified using the payload signature-based approach. Note that we intentionally mixed YouTube traffic with HTTP traffic used for web browsing to check whether our method can classify YouTube traffic from normal web traffic correctly. Table 2 shows a part of the HTTP and YouTube signaling packets. The similarity between the first packets of HTTP flow and YouTube flow is relatively high, about 0.79. Thus, it is difficult to decide a clear boundary between HTTP traffic and web-based application traffic, such as video streaming and web-storage.

We compare the overall accuracy of proposed method with that of LASER [4], which relies on application-level signatures to classify the traffic (Table 3). The overall accuracy of computing flow similarity improves slightly compared to LASER. However, the false negatives for each application are lower than LASER except Fileguri due to some packet loss while capturing.

Table 1. Classification accuracy using TMA

Application	**Classified Traffic (kB)**	**False Negative (kB)**	**False Positive (kB)**	**Overall Accuracy (%)**
BitTorrent	202,018	3,361	0	98.36
LimeWire	87,678	2,951	0	96.74
Fileguri	95,804	9,691	0	90.81
YouTube	16,061	0	3,775	80.96
TMA Log Traffic	421,339 kB			

Table 2. HTTP packets vs. YouTube signal packets

HTTP packet contents	**YouTube signal packet contents**
GET / HTTP/1.1 User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) ... ... Connection: Keep-Alive	GET /videoplayback?sparams=id%2C expire%2Cip%2Cipbits%.... ...... HTTP/1.1 User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) ...... Connection: Keep-Alive

Table 3. Flow similarity approach (FSA) vs. LASER [4]

Application	**Approach**	**False Negative (%)**	**False Positive (%)**
BitTorrent	FSA	1.66	0
	LASER	10.40	0
LimeWire	FSA	3.36	0
	LASER	8.42	0
Fileguri	FSA	10.11	0
	LASER	0.39	0
Overall Accuracy	LASER	97.39%	
	FSA	96.01%	

Finally, we estimate the population proportion with 95% confidence. The confidence interval is (6). The estimated classification accuracy is 96.01±0.0011.

$$\hat{p} - z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}} < p \le \hat{p} + z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}, \quad (6)$$

where $z_{\alpha/2} = 1.96$.

5 Conclusion

This paper proposes a traffic classification approach based on payload vector similarity. We suggest converting payloads into vector representations. Then, we apply a variation of simple document classification approach to classify traffic classification. For validation, we classify several target application traffic from our campus network traffic. The overall classification accuracy shows about 96%. Web-based application traffic is also identified, but there exists a small portion of false positives because it is very similar to HTTP traffic.

The proposed methods are based on the similarity between payload vectors, which is a metric of defining how similar flows are to each other. The exhaustive task of extracting the payload signatures from traffic is not required, and we can represent the similarity between payloads or flows as a simple numerical value. Our approach shows a reasonable performance even when high-volume applications, such as P2P traffic and web-based video streaming, are being measured.

For future work, we will increase the number of applications and develop a real-time agent program. We also plan to perform the same experiments using other similarity metrics.

References

[1] IANA, IANA port number list, `http://www.iana.org/assignments/port-numbers/`

[2] Moore, A.W., Papagiannaki, K.: Toward the Accurate Identification of Network Applications. In: Passive and Active Measurement Conference, Boston, MA, USA, March 31-April 1 (2005)

[3] Karagiannis, T., Papagiannaki, K., Faloutsos, M.: BLINC: Multilevel Traffic Classification in the Dark. In: ACM SIGCOMM 2005, Philadelphia, PA, USA, August 21-26 (2005)

[4] Park, B., Won, Y.J., Kim, M.-S., Hong, J.W.: Towards Automated Application Signature Generation for Traffic Identification. In: IEEE/IFIP Network Operations and Management Symposium (NOMS 2008), Salvador, Brazil, April 7-11, pp. 160–167 (2008)

[5] Karagiannis, T., Broido, A., Brownlee, N., claffy, K., Faloutsos, M.: Is P2P Dying or just Hiding? In: IEEE Globecom 2004, Dallas, Texas, USA, November 29-December 3 (2004)

[6] Kim, S.S., Reddy, A.L.N.: Image-Based Anomaly Detection Technique: Algorithm, Implementation and Effectiveness. IEEE Journal of Selected Areas in Communications 24, 1942–1954 (2006)

[7] Haffner, P., Sen, S., Spatscheck, O.: ACAS: Automated Construction of Application Signatures. In: ACM SIGCOMM 2005, Philadelphia, PA, USA, August 21-26 (2005)

[8] Sen, S., Spatscheck, O., Wang, D.: Accurate, Scalable In-Network Identification of P2P Traffic using Application Signatures. In: International World Wide Web Conference, NY, USA, May 19-21, pp. 512–521 (2004)

[9] Moore, A.W., Zeuv, D.: Internet Traffic Classification Using Bayesian Analysis Techniques. In: International Conference on Measurements and Modeling of Computer Systems, Banff, Alberta, Canada, June 6-10, pp. 50–60 (2005)

[10] Erman, J., Mahanti, A., Arlitt, M., Williamson, C.: Identifying and Discriminating Between Web and Peer-to-peer Traffic in the Network Core. In: International World Wide Web Conference, Banff, Alberta, Canada, May 8-12, pp. 883–892 (2007)

[11] Erman, J., Arlitt, M., Mahanti, A.: Traffic Classification Using Clustering Algorithms. In: SIGCOMM Workshop on Mining Network Data, Pisa, Italy, September 11-15, pp. 281–286 (2006)

[12] Karagiannis, T., Broido, A., Faloutsos, M., claffy, K.: Transport Layer Identification of P2P Traffic. In: Internet Measurement Conference, Taormina, Sicily, Italy, October 25-27, pp. 121–134 (2004)

[13] Choi, T.S., Kim, C.H., Yoon, S., Park, J.S., Lee, B.J., Kim, H.H., Chung, H.S., Jeong, T.S.: Content-aware Internet Application Traffic Measurement and Analysis. In: IEEE/IFIP Network Operations and Management Symposium (NOMS 2004), Seoul, Korea, April 23, vol. 1, pp. 511–524 (2004)

[14] Gummadi, K.P., Dunn, R.J., Saroiu, S., Gribble, S.D., Levy, H.M., Zahorjan, J.: Measurement, Modeling, and Analysis of a Peer-to-Peer Filesharing Workload. In: ACM Symposium on Operating Systems Review, December 2003, vol. 27, pp. 314–329 (2003)

[15] Salton, G., Buckley, C.: Term-weighting Approaches in Automatic Text Retrieval. Information Processing and Management 24(5), 513–523 (1988)

[16] Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (2008)

[17] Luhn, H.P.: A Statistical Approach to the Mechanized Encoding and Searching of Literary Information. IBM Journal of Research and Development, 309–317 (October 1957)

[18] Iliofotou, M., Pappu, P., Faloutsos, M., Mitzenmacher, M., Singh, S., Varghese, G.: Network monitoring using traffic dispersion graphs. In: Internet Measurement Conference, San Diego, CA, USA, October 24-26 (2007)

[19] BitTorrent, `http://www.bittorrent.com/`

[20] Emule, `http://www.emule-project.net/`

[21] YouTube, `http://youtube.com/`

[22] Fileguri, `http://www.fileguri.com/`

[23] Afreeca, `http://www.afreeca.com/`

[24] Endace, DAG 4.3GE, `http://www.endace.com/`