Frontiers of Information Technology & Electronic Engineering
www.zju.edu.cn/jzus; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Fine-grained P2P traffic classification by simply counting flows*

Jie HE[‡1], Yue-xiang YANG[1], Yong QIAO[2], Wen-ping DENG[1]

([1]*College of Computer, National University of Defense Technology, Changsha 410073, China*)

([2]*The Research Institution, China Electronic Equipment & System Engineering Company, Beijing 100141, China*)

E-mail: hejie@nudt.edu.cn; yyx@nudt.edu.cn; brave_jo@163.com; wpdeng.nudt@gmail.com

Received July 22, 2014; Revision accepted Dec. 12, 2014; Crosschecked Apr. 9, 2015

Abstract: The continuous emerging of peer-to-peer (P2P) applications enriches resource sharing by networks, but it also brings about many challenges to network management. Therefore, P2P applications monitoring, in particular, P2P traffic classification, is becoming increasingly important. In this paper, we propose a novel approach for accurate P2P traffic classification at a fine-grained level. Our approach relies only on counting some special flows that are appearing frequently and steadily in the traffic generated by specific P2P applications. In contrast to existing methods, the main contribution of our approach can be summarized as the following two aspects. Firstly, it can achieve a high classification accuracy by exploiting only several generic properties of flows rather than complicated features and sophisticated techniques. Secondly, it can work well even if the classification target is running with other high bandwidth-consuming applications, outperforming most existing host-based approaches, which are incapable of dealing with this situation. We evaluated the performance of our approach on a real-world trace. Experimental results show that P2P applications can be classified with a true positive rate higher than 97.22% and a false positive rate lower than 2.78%.

Key words: Traffic classification, Peer-to-peer (P2P), Fine-grained, Host-based
doi:10.1631/FITEE.1400267 **Document code:** A **CLC number:** TP393

1 Introduction

In recent years, statistical studies on Internet traffic have outlined that peer-to-peer (P2P) file-sharing applications still account for a large part of Internet traffic. For example, during the first half of 2014, P2P file-sharing traffic accounted for around 29% of the total Internet traffic in Asia-Pacific area (Sandvine, 2014). Appropriate network management, resource optimization, and intrusion detection can be performed only when P2P traffic is identified with high accuracy. However, characterizing and classifying P2P traffic is still a great challenge due to both the large number of newly emerging P2P protocols and their intentional use of random port numbers and encryption for communication.

Currently, there are mainly four types of approach in traffic classification according to application protocols (Gomes *et al.*, 2013). First, traditional port-based classification is a simple approach built upon the assumption that applications use their standard port numbers assigned by the Internet Assigned Numbers Authority (IANA). However, 90% of the modern P2P traffic may be using random ports (Basher *et al.*, 2008). The second type, deep packet inspection (DPI), is based on the inspection of packet payload. These methods can usually achieve high accuracy, but their drawbacks are well-known. They are resource-consuming and time-expensive, and thus often unfeasible in high-speed networks.

[‡] Corresponding author

[*] Project supported by the National Natural Science Foundation of China (Nos. 61170286 and 61202486)

ORCID: Jie HE, http://orcid.org/0000-0003-2244-7594

In addition, they cannot handle encrypted traffic. To address these challenges, many studies have proposed alternative mechanisms based on statistical information of the transport layer. Such approaches, also referred to as 'in the dark' (Karagiannis *et al.*, 2005), classify the traffic by statistical features extracted from host behaviors or transport layer data, including the sum of neighbor hosts connected in time window, variance of packet sizes in a flow, etc. The methods for classification in the dark are independent of the port number and payload, and always achieve high accuracy. However, they are growing in complexity, compromising one of their main motivations (Gomes *et al.*, 2013). The aforementioned approaches are all passive. Some researchers have proposed active crawlers for traffic classification (Ohzahata *et al.*, 2005). This kind of method is generally adopted for very constrained purposes, such as the identification of hosts running a specific P2P application.

In this paper, we aim to develop a simple, effective, and fine-grained P2P traffic classifier. Our approach aims to identify which host is running a P2P client and what it is, based solely on counting special flows generated by hosts within given time windows. These 'special' flows are defined as the most frequent and steady flows in the corresponding P2P traffic, which constitute strong evidence of existence of corresponding P2P applications. Different from previous in-the-dark approaches, this classifier exploits only generic properties of flows, and it does not require any complicated behavior features or traffic statistical features. Furthermore, unlike previous host-based classifiers, our engine is capable of identifying P2P hosts within complex traffic. We evaluate our classifier on real traffic traces collected from our campus network. Experimental results show that our classifier achieves good performance, even in the presence of multiple P2P clients running on the same host.

2 Related work

2.1 Deep packet inspection based approaches

Traditional DPI-based approaches, which rely on signatures for specific applications, can identify only traffic generated by these applications, and will become incapable when the traffic is encrypted. To eliminate these limitations, some new DPI-based methods are emerging (Dhamankar and King, 2007; Finamore *et al.*, 2010; Hullár *et al.*, 2011), which use the payload data from different perspectives. Dhamankar and King (2007) used entropy to explore the randomness of the encrypted payloads of Skype traffic. Some other researchers have studied learning relevant protocol patterns automatically from traffic traces. KISS (Finamore *et al.*, 2010) is a classifier that automatically extracts statistical signatures from User Datagram Protocol (UDP) streams by means of a Chi-square like test, which allows classification of the application protocol 'format', while ignoring the synchronization and semantic rules. The authors tested the mechanism using traffic traces from the real world, and obtained an excellent result. The average true positive rate was 99.6% and the average false positive rate was 0.4%. Hullár *et al.* (2011) addressed the classification of P2P applications using the first 16 bytes of payload of the first few packets of each flow, by applying techniques including the Markov model and random forests. Although these new DPI-based approaches work well with encrypted traffic, they still need sophisticated algorithms and inspection of payload information, which inevitably brings about excessive computation cost.

2.2 In-the-dark approaches

According to the granularity of classification objects (Dainotti *et al.*, 2012), the in-the-dark approaches can be roughly divided into two categories. The first type, which is known as flow-based classification, classifies traffic based on statistical features extracted from flows or bidirectional flows, such as packets inter-arrival time, flow duration, and idle time (Huang *et al.*, 2008; Chen, 2011; Tabatabaei *et al.*, 2012). These approaches ascribe flow objects to different applications. Moore *et al.* (2005) have listed a comprehensive set with 249 flow-level statistical features, intended to provide a reference for the community. Este *et al.* (2009) focused on the stability of the information carried by these flow-level features.

Another type can be summarized as host-based approaches, which classify a host by the predominant traffic it generates. This method aims to identify hosts running certain applications and provide results at the host level. Unlike flow-based classification, the objects from which statistical features are extracted usually include all traffic generated by a host, instead of flows. These host-level features reveal the social behaviors (also referred to as behavioral patterns) of the hosts. BLINC is a significant host-based classifier presented by Karagiannis *et al.* (2005). It analyzes patterns of host behavior at three levels (social, functional, and application), and extracts behavior features like the relation with other hosts, the role in the connection, and the transport layer information. BLINC is able to classify most traffic with a true positive rate ranging from 90% to 95%. Ban *et al.* (2012) presented another host-based mechanism to identify BitTorrent and PPLive. They revealed the behavior patterns of the hosts by employing entropy over source ports, TCP flags, source IPs, etc. Abacus (Bermolen *et al.*, 2011) is a new approach to identifying P2P-TV hosts by simply counting the number of packets generated by the hosts during short time windows and uses support vector machine (SVM) to train the mechanism. The experiments presented showed a true positive rate of 91.3%–99.6%, with a false positive rate of only 0.3%–8.7%. Similar to what has been done by Moore *et al.* (2005) and Este *et al.* (2009), Valenti and Rossi (2011) investigated a group of 109 host-level behavioral features for identifying P2P hosts, and assessed the stability of them by quantifying the amount of information contained in them.

Although the methods for in-the-dark classification usually work well with high accuracy, most of them (both flow-based and host-based) employ extra complicated techniques to extract features, such as entropy theory (Ban *et al.*, 2012), flow graph (Iliofotou *et al.*, 2011), and link homophily (Gallagher *et al.*, 2010). Moreover, a large part of them conduct traffic classification based on different supervised or unsupervised machine learning (ML) techniques, like SVM (Finamore *et al.*, 2010; Bermolen *et al.*, 2011; Tabatabaei *et al.*, 2012; He *et al.*, 2013), Bayesian (Auld *et al.*, 2007), and clustering (Nguyen and Armitage, 2008). Similarly, these sophisticated techniques and mechanisms require excessive computational and memory resources, which may make them unavailable in high-speed networks. On the other hand, approaches based on host behavior patterns often become ineffective when the host is in a complex network context. The host-level statistical features employed by them, such as the total number of destination IPs, variances of packet size, and mean payload length, will become invalid when the traffic is mixed with other unexpected applications, especially the high bandwidth-consuming ones, like video streaming and file hosting web services.

Our work can be ascribed to the host-based approaches with an important improvement. This study distinguishes itself from the aforementioned works by its simplification in classification mechanism and feasibility in a complex host network context. In the classification phase of our methodology, we are able to achieve a high classification accuracy by simply counting the number of some 'special' flows. Neither complicated statistical features nor sophisticated ML algorithms are needed.

3 Classification methodology

Our aim is to classify P2P hosts from ordinary hosts and identify which P2P applications are running on them. In this paper, we focus mainly on P2P file-sharing applications, and choose five popular platforms for evaluation, including BitComet (BC), BitTorrent (BT), eMule (EM), Vagaa (VG), and Thunder (TD). Other categories of P2P applications will be addressed in the near future. Our classification methodology is a two-phase process. Firstly, we define 'special' flows that can significantly represent the existence of corresponding P2P applications and highlight them for each P2P application. After that, we conduct traffic classification with the help of these 'special' flows.

3.1 Definition of 'special' flows

Different P2P applications prefer different transport layer protocols to communicate with each other and transfer data. Therefore, we concentrate on both UDP traffic and Transmission Control Protocol (TCP) traffic. The whole traffic generated by host H is captured and collected to be a set of flow records. In this study, a flow is defined as a set of IP packets sharing the same source, destination IP addresses, associated port numbers, and protocols. A 5-tuple is employed to represent flows, namely $\langle \mathrm{Proto}, \mathrm{IP}_{\mathrm{src}}, \mathrm{Port}_{\mathrm{src}}, \mathrm{IP}_{\mathrm{dst}}, \mathrm{Port}_{\mathrm{dst}} \rangle$. A flow is considered expired when:

1. The flow is inactive for a certain time period (no new packets received for the flow), which is set

to 10 min in this study.

2. The flow is long lived (active) and lasts longer than the active timer, which is set to 30 min.

3. A TCP flag that indicates the termination of the flow has been seen, i.e., FIN, RST flag.

We restrict our attention to successful flows, which have completed SYN, SYN/ACK, and ACK handshakes in the TCP case, or completed at least one packet exchanging in the UDP case.

To keep the effectiveness and robustness of P2P networks, each host of a P2P network has to exchange a lot of signal packets (e.g., peer discovery packets, content request packets, notification packets) periodically or frequently with other hosts in the same P2P network (Yang *et al.*, 2009). Although abundant analogous signal activities exist in all P2P networks, there are some inherent differences among them due to the dissimilarity of P2P protocols. In previous work, complicated features like ratio, variance, and even entropy of flow-level or host-level statistical features, were employed to highlight these differences. In contrast, we rely solely on several basic properties of flows that can be efficiently computed.

We notice that usually there are a large amount of flows with similar characteristics (e.g., the amount and size of packets) in the signal traffic generated by P2P hosts. Furthermore, the characteristics of these similar flows are the same for hosts in the same P2P network, and vary for hosts using different P2P protocols. In other words, if two flows are generated by the same P2P application and correspond to the same signal activity, they tend to have the same transport layer protocol, packet amount, and size. There may be a lot of groups of similar flows in the traffic generated by a P2P application, since a number of different signal activities are performed. Therefore, we concentrate only on the signal flows, and exclude 'long' flows which are usually used to transmit data chunks in a P2P network (Hurley *et al.*, 2011). Additionally, to obtain more download resources and faster download speed, each host has to send out a lot of signal flows due to their 'client' characteristic in a P2P network. Therefore, we consider only outgoing traffic from hosts. The remaining traffic is collected for each host H within the monitored network, referred to as $F(H)$.

To make a temporal estimate of the distribution of the flows after reduction, we collect and examine flows generated by BC and EM during 1 h. The results are shown in Fig. 1.

Fig. 1a shows that 45.85% flows generated by BC in 1 h are UDP flows, which send only one packet with 140 bytes and receive one packet with 333 bytes. The same phenomenon occurs in the traffic generated by EM (Fig. 1b), where 28.63% of the flows send only an 85-byte UDP packet and receive an 86-byte UDP packet. Fig. 1 implies that a large part of the flows generated by P2P applications share similar amount and size of packets, and this phenomenon varies for different P2P applications.

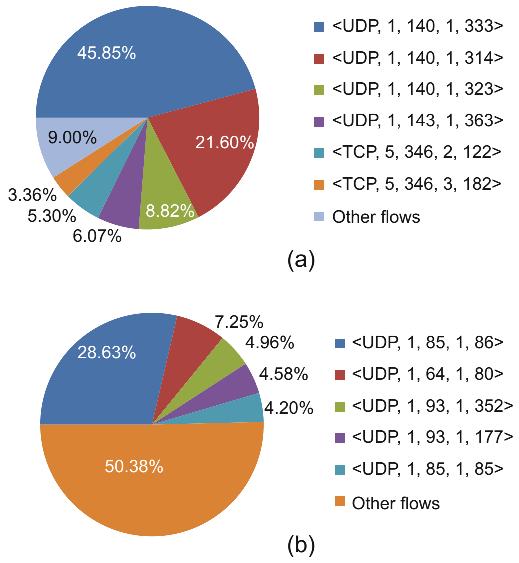


Fig. 1 Distribution of similar flows in BitComet (a) and eMule (b). References to color refer to the online version of this figure

Each flow generated by host H is described as a vector $\boldsymbol{v}(H)$ with five generic elements, namely $\langle P, S_{\text{pkts}}, S_{\text{bts}}, R_{\text{pkts}}, R_{\text{bts}}\rangle$, in which P represents the transport layer protocol of the flow, S_{pkts} and S_{bts} represent the amount and size of the packets being sent respectively, and R_{pkts} and R_{bts} represent the amount and size of the packets received respectively. Then the flow set $F(H)$ generated by host H can be expressed as a set of flow vectors $V(H) = \{\boldsymbol{v}(H)_i | i = 1, 2, ..., |F(H)|\}$.

To partition $V(H)$ into clusters of similar flow vectors, we apply BIRCH (Zhang *et al.*, 1996) as a clustering algorithm, which is an efficient data clustering method for very large databases. Each of the sub-clusters of flow vectors $C_{q(H)}$ output by BIRCH, represents a group of flows with similar protocol, packet amount and size. When the hourly amount of flows in $C_{q(H)}$ is larger than a threshold m, we consider this group of flows as belonging to a certain type of signal activity. For each of these sub-clusters, we aggregate flows in it and represent it using an average vector, which we name 'clustering flow', **CF** for short:

$$\mathbf{CF}_q = \left\langle P, \overline{S_{\text{pkts}}}, \overline{S_{\text{bts}}}, \overline{R_{\text{pkts}}}, \overline{R_{\text{bts}}} \right\rangle, \qquad (1)$$

where P is the protocol of the flows in the sub-cluster, and the other four elements are computed by averaging values of corresponding properties in the sub-cluster. In this way, we can obtain a set of **CF** for every P2P application, denoted by CF_{P2P} as follows:

$$\text{CF}_{\text{P2P}} = \Big\{ \mathbf{CF} \Big| \mathbf{CF}_q = \left\langle P, \overline{S_{\text{pkts}}}, \overline{S_{\text{bts}}}, \overline{R_{\text{pkts}}}, \overline{R_{\text{bts}}} \right\rangle \ \& \ \left| C_q(H) \right| / h > m \Big\}, \qquad (2)$$

where h is the time measured by hour. In other words, CF_{P2P} is a brief summary of major signal activities of this P2P network.

Fig. 2 provides a pictorial representation of the flow clustering process of a P2P host which is performing two kinds of signal activities. Flows corresponding to peer discovery and notification are grouped into two sub-clusters ($C_1(H)$ and $C_2(H)$), due to their similar properties (protocol, packet size and number).

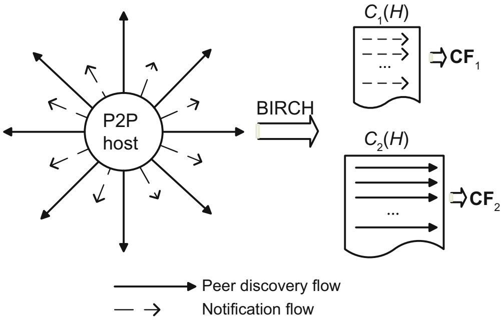


Fig. 2 Flow clustering for P2P host

3.2 Classification mechanism

After extracting CF_{P2P} for each P2P application, we can identify their existence by simply counting the number of **CF**. We define CF_{set} as the set of all **CF** extracted from P2P applications at the previous phase, i.e., $\text{CF}_{\text{set}} = \bigcup \text{CF}_{\text{P2P}}$. Recall that our aim is to identify which host is running a P2P application and what it is. To this end, we monitor the network traffic generated by host H and count the number of every **CF** appearing in time window T, which is recorded as N_q, where $q = 1, 2, ..., |\text{CF}_{\text{set}}|$. As **CF** is an average vector of flow vectors in $C_{q(H)}$, we consider all approximate $\boldsymbol{v}(H)$ included in Eq. (3) as **CF**, i.e.,

$$\boldsymbol{v}(H) \in \Big\{ \left\langle P, \overline{S_{\text{pkts}}}, \overline{S_{\text{bts}}} \pm \lambda_1, \overline{R_{\text{pkts}}}, \overline{R_{\text{bts}}} \pm \lambda_2 \right\rangle \Big| \ \lambda_1 = \left\lfloor \overline{S_{\text{bts}}}/100 \right\rfloor, \lambda_2 = \left\lfloor \overline{R_{\text{bts}}}/100 \right\rfloor \Big\}. \qquad (3)$$

S_{pkts} and R_{pkts} of $\boldsymbol{v}(H)$ need to precisely equal $\overline{S_{\text{pkts}}}$ and $\overline{R_{\text{pkts}}}$ of **CF** respectively, since the number of packets of **CF** is always small. λ_1 and λ_2 represent the approximate range of the size of all packets, which we conservatively set to $\left\lfloor \overline{S_{\text{bts}}}/100 \right\rfloor$ and $\left\lfloor \overline{R_{\text{bts}}}/100 \right\rfloor$ ('$\lfloor \cdot \rfloor$' means the corresponding value is rounded down).

Then a score function is defined to estimate whether host H is running certain P2P applications or not. The score function is depicted as

$$\text{Score}_{\text{P2P}} = \sum_{q'=1}^{|\text{CF}_{\text{P2P}}|} \alpha_{q'} N_{q'}, \qquad (4)$$

where $N_{q'}$ is the appearance time of the corresponding $\mathbf{CF}_{q'}$ in CF_{P2P}, and $\alpha_{q'}$ is the weight of $\mathbf{CF}_{q'}$, $q' = 1, 2, \ldots, |\text{CF}_{\text{P2P}}|$. Apparently, the more frequent and steady $\mathbf{CF}_{q'}$ is, the more important it is. The assignation of $\alpha_{q'}$ will be discussed in Section 4.3.1.

For every P2P application, this score function will be calculated once to obtain its score value $\text{Score}_{\text{P2P}}$ at the end of time window T. When $\text{Score}_{\text{P2P}}$ is greater than a certain threshold S_{P2P}, we draw the conclusion that host H is running this P2P application. The value of threshold S_{P2P} will be discussed in Section 4.3.2. If none of the five P2P applications is detected in time window T, we consider that the traffic generated by host H belongs to other applications.

The overall workflow of our classification mechanism is depicted in Fig. 3. In phase 1, representative traces of P2P applications are collected and further reduced by filtering out unconcerned flows. The remaining flows, expressed as flow vectors, are fed into BIRCH to extract **CF** for this P2P application. In the classification phase, the traffic of a suspected host is monitored to detect and count the appearance of **CF** extracted in the first phase. After every time window T, the decision module will make a classification decision according to the values of the score function.

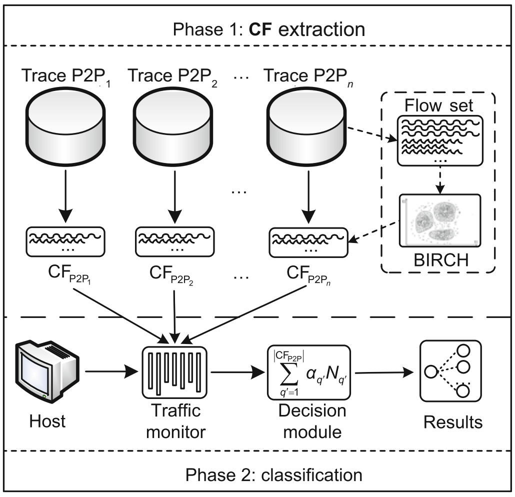


Fig. 3 Workflow of the classification mechanism

The clustering process of BIRCH may be a slow and computation-intensive process. However, it is performed only once for every P2P trace in the **CF** extraction phase rather than the classification phase. Once all **CF** are extracted from traces, no more clustering processes are needed. Besides, no machine learning algorithm is employed in phase 2. All we need in the classification phase are several generic properties of flow records and a score function. Thus, our classification mechanism supports real-time processing.

4 Dataset and parameter selection

4.1 Dataset collection

To evaluate our classification methodology, we choose five popular P2P file-sharing applications, including BitComet (BC), BitTorrent (BT), eMule (EM), Vagaa (VG), and Thunder (TD). We assume that it is easy to distinguish the difference among applications based on different P2P protocols, but relatively difficult to tell the difference among applications based on the same P2P protocol. Therefore, to demonstrate the efficacy of our classification methodology to the greatest extent, we intentionally choose several P2P applications based on the same protocols. For example, BC and BT are different implementations of the BitTorrent protocol, while EM and VG are both based on the eDonkey protocol. Nevertheless, the range of the P2P applications we could classify is not limited to the five samples. The characteristic of the flow clustering behavior is common in all P2P applications due to their inherent nature. Therefore, our approach has good generality and can be easily applied to classify other P2P applications.

Two datasets are collected in our experiments. First, training traces are collected in a completely operational network, and used to extract CF_{P2P} for each P2P application. The other dataset is the real-world traffic trace collected from our campus network, which is used to evaluate the performance of our classification method.

4.1.1 Collection of training traces

To extract **CF** for the five P2P applications, we first collect training traces in a fully controlled environment. We set up five Windows XP virtual machines with public IP addresses, and separately run each of the five P2P applications on them for a long period (7 d). By means of AutoIt scripts (Autoit Consulting Ltd., England), the five hosts automatically and randomly select content to download or upload at random time intervals, using the P2P applications running on them. All traffic generated by them is captured and collected in the unit of flow with a C program we have developed, named Traffic Logger. Table 1 summarizes the five training traces and reports brief information of them. Using a trace from a fully controlled environment has the advantage of providing a reliable ground truth, as there is no doubt on the application generating the traffic.

4.1.2 Collection of real-world traces

We evaluate the performance of our classification mechanism using the real-world traffic trace collected from our campus network. This comes from a span port mirroring all traffic crossing the gateway router for the campus network. ‘Traffic Logger’ is used to collect flow-level trace for every running host within the campus network for 24 h. Overall, we observe 315 active hosts in the campus network.

Establishing the ground truth (i.e., what is the actual application that generates the traffic), is a crucial and difficult part of traffic classification studies. Some previous work obtained the ground truth using existing DPI tools. Unfortunately, the results may be unreliable due to the shortcomings of DPI engines. So, we obtain the ground truth by manually investigating each of these hosts. Through manual validation, we identify 8 BC hosts, 2 BT hosts, 21 EM hosts, 13 VG hosts, and 56 TD hosts. Brief information on these hosts is given in Table 2. All other traffic that is not generated by the five P2P applications, such as web surfing, online games, and video streaming, is defined as background (BG) traffic.

Table 1 Description of training traces

Trace	Duration (d)	Number of packets ($\times10^6$)	Sum of flows ($\times10^6$)	Sum of successful UDP flows ($\times10^3$)	Sum of successful TCP flows ($\times10^3$)
BC	7	610.73	5.21	2873.00	44.20
BT	7	1397.95	1.67	250.27	116.98
EM	7	1398.33	0.35	105.96	114.82
VG	7	631.39	0.23	55.37	42.24
TD	7	629.36	2.33	144.93	157.51

BC: BitComet; BT: BitTorrent; EM: eMule, VG: Vagaa; TD: Thunder

Table 2 Description of real-world traces

App.	Sum of hosts	Active time (h)	Sum of flows ($\times10^6$)	Sum of successful UDP flows ($\times10^3$)	Sum of successful TCP flows ($\times10^3$)
BC	8	48	1.47	666.75	12.48
BT	2	14	0.12	19.24	6.06
EM	21	84	0.34	84.77	105.23
VG	13	104	0.25	43.34	37.15
TD	56	840	11.14	1424.24	1114.24
BG	215	1720	32.85	9173.65	15552.74

App.: application. BC: BitComet; BT: BitTorrent; EM: eMule; VG: Vagaa; TD: Thunder; BG: background

4.2 CF extraction

To extract $\mathrm{CF_{P2P}}$ for P2P applications, we undertake the following steps. First, we reduce the volume of the five training traces by excluding all other flows except successful UDP and TCP flows. Then long flows, whose S_{pkts} or R_{pkts} is larger than 100, are filtered out. After that, we apply a BIRCH clustering algorithm on the reduced traces, with the help of Rstudio with the BIRCH package. The main clustering results are illustrated in Fig. 4. For lack of space, we just present the top 10 sub-clusters for each P2P application. It is apparent from this figure that all the five applications have generated many groups of similar flows. To facilitate the evaluation, we aggregate sub-clusters whose hourly average number is greater than m, which we set to 10, to an average vector **CF** as mentioned in Section 3.1, i.e.,

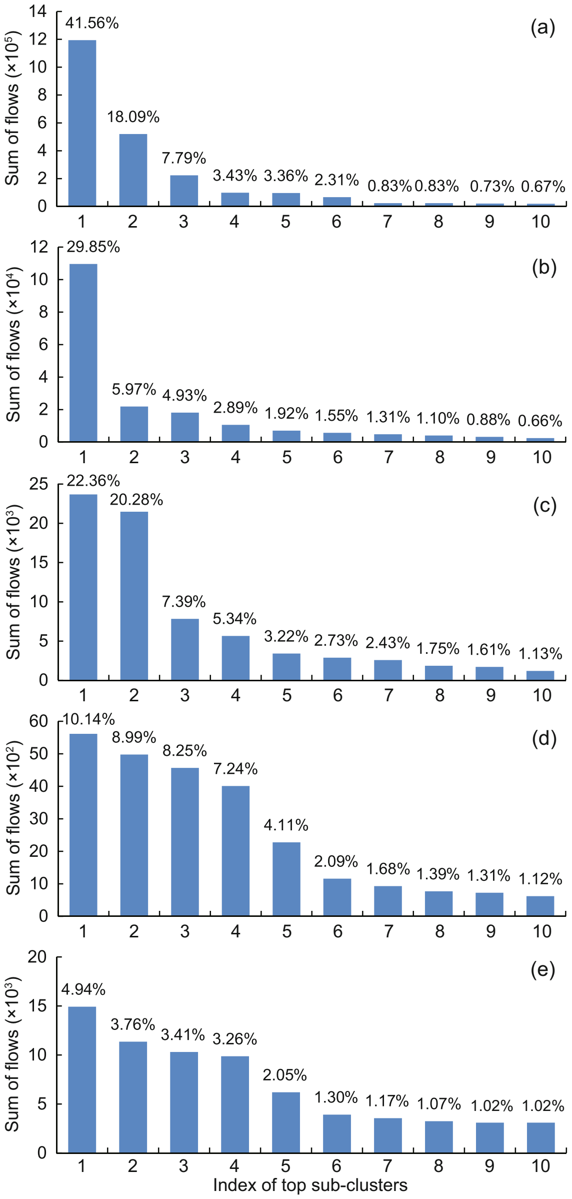


Fig. 4 Top 10 sub-clusters of training traces: (a) BC; (b) BT; (c) EM; (d) VG; (e) TD

$$\mathrm{CF_{P2P}} = \left\{ \mathbf{CF} \middle| \mathbf{CF}_q = \langle P, \overline{S_{\mathrm{pkts}}}, \overline{S_{\mathrm{bts}}}, \overline{R_{\mathrm{pkts}}}, \overline{R_{\mathrm{bts}}} \rangle \right.$$
$$\left. \& \ |C_q(H)| \,/ (7 \times 24) > 10 \right\}. \tag{5}$$

We extract $\mathrm{CF}_{\mathrm{BC}}$, $\mathrm{CF}_{\mathrm{BT}}$, $\mathrm{CF}_{\mathrm{EM}}$, $\mathrm{CF}_{\mathrm{VG}}$, and $\mathrm{CF}_{\mathrm{TD}}$ from the five training traces, respectively. The summary information of each $\mathrm{CF}_{\mathrm{P2P}}$ is given in Table 3. For the sake of brevity, not all **CF** are reported here.

We notice that some applications are sharing several common **CF**. For example, $\langle \mathrm{TCP}, 5, 346, 3, 182\rangle$ is shared by BC, BT, and TD, while $\langle \mathrm{UDP}, 1, 64, 1, 80\rangle$ is shared by both EM and VG. The possible reason is that both BC and BT are developed based on the BitTorrent protocol, while TD is also developed to support the BitTorrent protocol, and both EM and VG are implemented based on the eDonkey protocol. Therefore, parts of their signal activities exhibit similarity in frequency and size of packets. Another interesting finding is that all $\mathrm{CF}_{\mathrm{P2P}}$ are dominated by UDP **CF**. A reasonable explanation for this may be that UDP is becoming a preferred transport layer protocol for P2P applications.

We further observe that the payload of flows corresponding to some selected **CF** uses the Wireshark. Table 4 shows that both of the dominating **CF** in $\mathrm{CF}_{\mathrm{BC}}$ and $\mathrm{CF}_{\mathrm{BT}}$, i.e., $\langle \mathrm{UDP}, 1, 140, 1, 333\rangle$ and $\langle \mathrm{UDP}, 1, 145, 1, 329\rangle$, represent flows for the peer discovery activity of the BitTorrent protocol. In other words, although BC and BT are implemented based on the same protocol, subtle differences still exist between their common signal activities. Nevertheless, we can distinguish these differences using **CF**.

There is a possibility that some **CF** of certain applications may change when this application updates its release, since the new release might have made some adjustments in its protocol. However, the inherent characteristics of flow clustering behavior will not change even when an application is updated. To resolve this problem, we can re-extract $\mathrm{CF}_{\mathrm{P2P}}$ for this application once its classification accuracy declines sharply.

4.3 Parameter selection

4.3.1 Assignment of weight $\alpha_{q'}$

As stated previously, $\alpha_{q'}$ is the weight of the corresponding $\mathbf{CF}_{q'}$ in the score function. The more frequent and steady $\mathbf{CF}_{q'}$ is, the more important it should be, i.e., the larger $\alpha_{q'}$ should be. To estimate the frequency and stability of **CF**, we count the hourly quantity of every **CF** in the

Table 3 Summary of $\mathbf{CF}_{\mathbf{P2P}}$

	$\mathrm{CF}_{\mathrm{BC}}$	$\mathrm{CF}_{\mathrm{BT}}$	$\mathrm{CF}_{\mathrm{EM}}$	$\mathrm{CF}_{\mathrm{VG}}$	$\mathrm{CF}_{\mathrm{TD}}$
CF	$\langle \mathrm{UDP}, 1, 140, 1, 333\rangle$	$\langle \mathrm{UDP}, 1, 145, 1, 329\rangle$	$\langle \mathrm{UDP}, 1, 85, 1, 86\rangle$	$\langle \mathrm{UDP}, 1, 80, 1, 86\rangle$	$\langle \mathrm{UDP}, 1, 143, 1, 363\rangle$
	$\langle \mathrm{UDP}, 1, 140, 1, 314\rangle$	$\langle \mathrm{UDP}, 1, 145, 1, 310\rangle$	$\langle \mathrm{UDP}, 2, 262, 2, 254\rangle$	$\langle \mathrm{UDP}, 2, 140, 2, 148\rangle$	$\langle \mathrm{UDP}, 1, 214, 1, 70\rangle$
	$\langle \mathrm{UDP}, 1, 140, 1, 323\rangle$	$\langle \mathrm{UDP}, 1, 145, 1, 319\rangle$	$\langle \mathrm{UDP}, 1, 93, 1, 177\rangle$	$\langle \mathrm{UDP}, 1, 93, 1, 352\rangle$	$\langle \mathrm{UDP}, 1, 143, 1, 331\rangle$
	$\langle \mathrm{UDP}, 1, 143, 1, 363\rangle$	$\langle \mathrm{UDP}, 1, 148, 1, 359\rangle$	$\langle \mathrm{UDP}, 2, 230, 1, 161\rangle$	$\langle \mathrm{UDP}, 1, 60, 1, 62\rangle$	$\langle \mathrm{UDP}, 1, 143, 1, 381\rangle$
	$\langle \mathrm{UDP}, 1, 104, 1, 114\rangle$	$\langle \mathrm{UDP}, 7, 541, 1, 65\rangle$	$\langle \mathrm{UDP}, 1, 262, 1, 177\rangle$	$\langle \mathrm{UDP}, 1, 64, 1, 80\rangle$	$\langle \mathrm{UDP}, 2, 286, 1, 363\rangle$
	$\langle \mathrm{UDP}, 1, 104, 1, 95\rangle$	$\langle \mathrm{UDP}, 1, 145, 1, 340\rangle$	$\langle \mathrm{UDP}, 1, 64, 1, 80\rangle$		$\cdots$
	$\cdots$	$\cdots$	$\langle \mathrm{UDP}, 1, 93, 1, 352\rangle$		$\langle \mathrm{TCP}, 5, 346, 3, 182\rangle$
	$\langle \mathrm{TCP}, 5, 346, 3, 182\rangle$	$\langle \mathrm{TCP}, 5, 346, 3, 182\rangle$	$\langle \mathrm{UDP}, 1, 85, 1, 85\rangle$		$\langle \mathrm{TCP}, 5, 402, 5, 596\rangle$
	$\langle \mathrm{TCP}, 5, 346, 4, 242\rangle$	$\langle \mathrm{TCP}, 5, 346, 4, 242\rangle$	$\langle \mathrm{UDP}, 1, 85, 1, 91\rangle$		$\langle \mathrm{TCP}, 5, 346, 4, 242\rangle$
	$\langle \mathrm{TCP}, 3, 238, 2, 138\rangle$	$\langle \mathrm{TCP}, 3, 238, 2, 138\rangle$			$\cdots$
SUM	50	12	9	5	18
UDP	47	9	9	5	11
TCP	3	3	0	0	7

Table 4 Payloads of some CF

App.	**CF**	Payload sent	Payload received	Signal activity
BC	$\langle \mathrm{UDP}, 1, 140, 1, 333\rangle$	d1:ad2:id20:…find_node1:…:y1:qe	d1:rd2:id20:…nodes208:…:y1:re	Peer discovery
	$\langle \mathrm{UDP}, 1, 104, 1, 114\rangle$	d1:ad2:id20:…:ping1:…:y1:qe	d1:rd2:id20:…:y1:re	Notification
	$\langle \mathrm{UDP}, 1, 143, 1, 363\rangle$	d1:ad2:id20:…info_hash20:…:y1:qe	d1:rd2:id20:…token20:…:y1:re	Content discovery
BT	$\langle \mathrm{UDP}, 1, 145, 1, 329\rangle$	d1:ad2:id20:…find_node1:…:y1:qe	d1:rd2:id20:…nodes208:…:y1:re	Peer discovery
TD	$\langle \mathrm{TCP}, 5, 346, 3, 182\rangle$	.BitTorrent protocol…Tl		Handshake

App.: application. BC: BitComet; BT: BitTorrent; TD: Thunder corresponding training trace, and calculate their means and standard deviations. Therefore, $\alpha_{q'}$ should be proportional to the mean and inversely proportional to the standard deviation. Furthermore, to remedy the imbalance of the sums of **CF** in the five CF_{P2P}, $\alpha_{q'}$ is standardized by setting $\sum_{q'=1}^{|\text{CF}_{\text{P2P}}|} \alpha_{q'} = 1$. In sum, we define $\alpha_{q'}$ as in Eq. (6), in which Mean_z and Stdev_z represent the mean and standard deviation of the hourly quantity of $\mathbf{CF}_z$ in the corresponding training trace, respectively.

$$\alpha_{q'} = \frac{\text{Mean}_{q'}/\text{Stdev}_{q'}}{\displaystyle\sum_{z=1}^{|\text{CF}_{\text{P2P}}|} \frac{\text{Mean}_z}{\text{Stdev}_z}}. \qquad (6)$$

4.3.2 Determination of threshold S_{P2P}

When the value of the score function $\text{Score}_{\text{P2P}}$ is greater than the threshold S_{P2P}, we consider that the monitored host is running this P2P application. Thus, S_{P2P} is an important metric in our method, which directly influences the classification accuracy.

By applying the score function $\text{Score}_{\text{P2P}}$ to the corresponding training trace, and calculating their score values in every time window T, which we set to 1 h, we obtain 168 scores for each application. Their minimum values are taken as S_{P2P} for each $\text{Score}_{\text{P2P}}$ for the following two reasons:

1. Since these five training traces are collected in a fully controlled environment, there is no doubt about the application generating the traffic. In other words, the scores calculated from them are all 'true positive'. Thus, it is reasonable to set the minimum score value as S_{P2P}.

2. We assume that our five training traces are representative of their general traffic. It is not only because of the large temporal span of our training traces, which include traffic generated during midday, midnight, weekend, etc., but also because of the universality of our P2P hosts' behaviors, which include searching, downloading, and uploading of a variety of types of content. To verify this assumption, we test the distribution of the scores stated above, and assess whether they meet the normal distribution, using quantile-quantile plots (Fig. 5). The slope of a straight line in the figure is the standard deviation of the score, while its intercept indicates the mean. The more the spread of points close to the straight line, the more the scores close to a normal distribution. It is apparent from these figures that all of them are basically in line with a normal distribution.

Consequently, we take the minimum score of the corresponding training traces as a threshold for each score function $\text{Score}_{\text{P2P}}$, that is, $S_{\text{P2P}} = \min_{\text{TrainingTrace}}\{\text{Score}_{\text{P2P}}\}_{T=1\text{ h}}$. Table 5 illustrates the values of S_{P2P}.

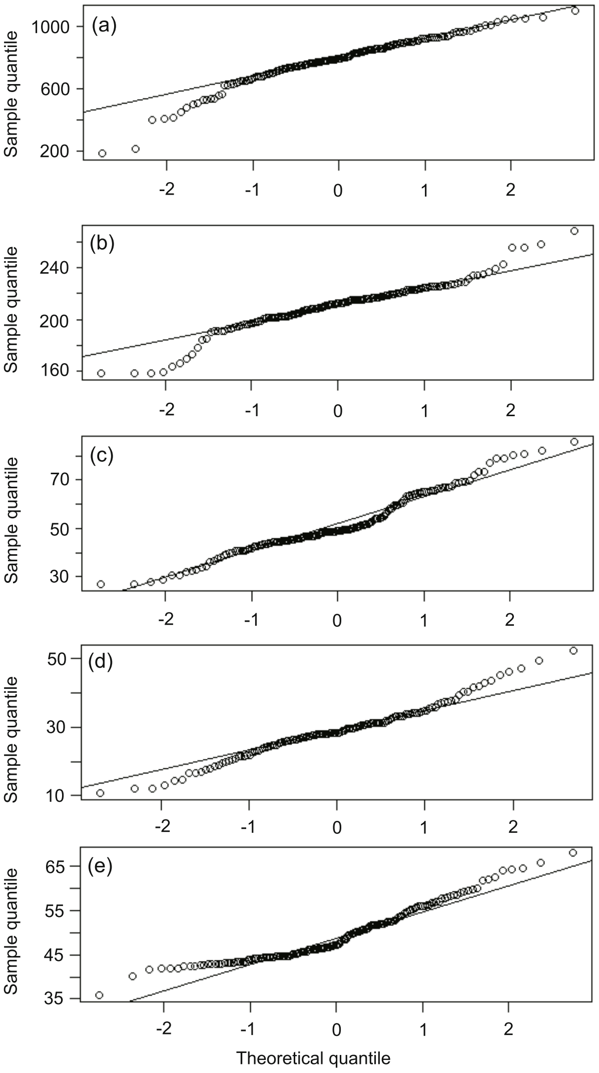


Fig. 5 Quantile-quantile plots for scores of training traces: (a) BC; (b) BT; (c) EM; (d) VG; (e) TD

Table 5 Value of each S_{P2P}

App.	BC	BT	EM	VG	TD
S_{P2P}	86.96	77.24	10.22	10.66	19.20

App.: application. BC: BitComet; BT: BitTorrent; EM: eMule; VG: Vagaa; TD: Thunder

5 Experimental results and analysis

The performance of our classification mechanism is evaluated with the real-world trace collected from our campus network. For every host in the monitored network, we draw a conclusion as to whether it is running a P2P application and what it is after every time window T, which we set to 1 h. To complete downloading, P2P applications usually run continuously for hours or even days. The traffic generated by short-lived ones is so small that we assume it has negligible impact on the whole network. Therefore, it is acceptable to set T to 1 h.

5.1 Metrics

We evaluate the classification performance in terms of the 'true positive rate (TPR)' and 'false positive rate (FPR)'. A sample is 'true positive (TP)' if it is correctly classified as belonging to a corresponding class. A sample is 'false positive (FP)' if it is incorrectly classified as belonging to the corresponding class. Similarly, a sample is said to be 'true negative (TN)' if it is correctly classified as not belonging to the corresponding class. A sample is said to be 'false negative (FN)' if it is incorrectly classified as not belonging to the corresponding class. Therefore, TPR and FPR are defined as follows:

$$\mathrm{TPR}=\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}},\ \mathrm{FPR}=\frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}}. \tag{7}$$

5.2 Results

The confusion matrix of classification results is illustrated in Table 6.

Table 6 Confusion matrix of classification results

	T.W	BC	BT	EM	VG	TD	BG
BC	48	**100%**	0	0	0	12.5%	0
BT	14	0	**100%**	0	0	0	0
EM	84	0	0	**100%**	5.95%	0	0
VG	104	0	0	11.54%	**99.04%**	0	0.96%
TD	840	0	0	0	0	**100%**	0
FPR	–	0	0	10.35%	5.67%	11.11%	–

T.W: total number of time windows. BC: BitComet; BT: BitTorrent; EM: eMule; VG: Vagaa; TD: Thunder; BG: background. FPR: false positive rate. Labels on the rows (except the last) represent the ground truth, and labels on the columns (except the second) represent the classification results. Bold values represent the true positive rate (TPR) of the corresponding application

The results are excellent. All five applications are detected in all time windows except one time window of VG. The TPR is always higher than 99.04%. Unfortunately, the FPR is not very satisfactory. 12.5% of BC time windows are classified into TD simultaneously, while 5.95% of VG and 11.54% of EM are mixed with each other, which leads to relatively high FPR of these applications (5.67% for VG, 10.35% for EM, and 11.11% for TD). One possible reason for this is that they share some common **CF**. Thus, a false positive occurs when the number of these common **CF** is large enough to meet the decision condition: $\mathrm{Score}_{\mathrm{P2P}} > S_{\mathrm{P2P}}$.

To solve this problem, we introduce the notion of KEY **CF**, which is defined as the unique and most significant (i.e., with the largest α) **CF** in each $\mathrm{CF}_{\mathrm{P2P}}$. The number of KEY **CF**, denoted by N_{key}, will be further checked after a time window once $\mathrm{Score}_{\mathrm{P2P}} > S_{\mathrm{P2P}}$. N_{key} should be close to 0 in other P2P traffic, since KEY **CF** is unique. Furthermore, KEY **CF** should appear frequently and steadily in its corresponding traffic, since it is the most significant **CF**. Therefore, after each time window, if the decision condition is met, but N_{key} is still smaller than n, which is set to 10, we consider that there is no such P2P traffic. The KEY **CF** of each P2P application is shown in Table 7.

Table 8 illustrates the results after introducing the KEY **CF** mechanism. From this table we can see that there is no longer any false positive among the five applications, which exactly demonstrates the excellent discrimination of our KEY **CF**. Meanwhile, all of the TPR remain at 100% except for a negligible decrease of VG. That is probable because the KEY **CF** of VG is not sufficiently steady, whose amount is less than n in a few time windows.

5.3 Dynamic time window

We notice that some P2P applications generate so many **CF** in 1 h that its score function gains a value much larger than its threshold S_{P2P}. To improve the classification efficiency, we employ a dynamic time window mechanism. That is, during a fixed length time window (1 h), we draw a conclusion that the host is running a certain P2P application as soon as the value of the score function of this P2P application is greater than its threshold (i.e., $\mathrm{Score}_{\mathrm{P2P}} > S_{\mathrm{P2P}}$) and the amount of its KEY **CF** is larger than the threshold n (i.e., $N_{\mathrm{key}} > n$). Then we open a new dynamic time window for the identification of this P2P application immediately. The new dynamic time window is solely used for identifying this P2P application, and parallels with the original fixed length time window, which will last till its end (1 hour) to identify other possible P2P applications. This mechanism could greatly accelerate the identification of P2P applications generating plenty of traffic, since there is no need to wait until the end of every whole time window.

Table 7 KEY CF of each P2P application

App.	KEY **CF**	Percentage* (%)	Sum per hour	α
BC	$\langle \mathrm{UDP}, 1, 140, 1, 333\rangle$	41.56	7108	0.06
BT	$\langle \mathrm{UDP}, 1, 145, 1, 329\rangle$	43.80	652	0.24
EM	$\langle \mathrm{UDP}, 1, 85, 1, 86\rangle$	22.36	141	0.25
VG	$\langle \mathrm{UDP}, 2, 140, 2, 148\rangle$	8.99	30	0.16
TD	$\langle \mathrm{UDP}, 1, 214, 1, 70\rangle$	6.81	60	0.66

App.: application. BC: BitComet; BT: BitTorrent; EM: eMule; VG: Vagaa; TD: Thunder. * Flow of KEY **CF**/Flow of P2P application

Table 8 Confusion matrix of classification results after introducing KEY CF

	T.W	BC	BT	EM	VG	TD	BG
BC	48	**100%**	0	0	0	0	0
BT	14	0	**100%**	0	0	0	0
EM	84	0	0	**100%**	0	0	0
VG	104	0	0	0	**98.07%**	0	1.93%
TD	840	0	0	0	0	**100%**	0
FPR	–	0	0	0	0	0	–

T.W: total number of time windows. BC: BitComet; BT: BitTorrent; EM: eMule; VG: Vagaa; TD: Thunder; BG: background. FPR: false positive rate. Bold values represent the true positive rate (TPR) of the corresponding application

The classification results and summary of dynamic time windows are illustrated in Fig. 6 and Table 9, respectively. Fig. 6 shows that the average classification time window is greatly reduced from 44.94% to 88.89% after employing the dynamic time window mechanism. In the best case, it takes only 8 s to identify an active BC host. Table 9 shows that the classification performance keeps excellent while the time cost is considerably reduced.

5.4 Classification in complex network context

Most of the existing host-based approaches for P2P traffic classification are based on behavioral features extracted from the whole traffic generated by hosts. These features are significant when the traffic is dominated by the classification target. Unfortunately, users of hosts do not usually run only one application at the same time. For example, they may surf the Internet and watch some videos at YouTube while waiting for P2P files to download. In this case, these behavioral features may lose effectiveness, due to the influence of unexpected traffic.

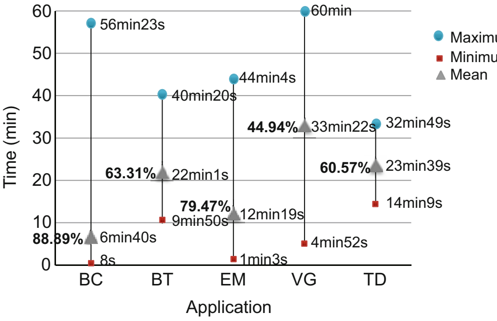


Fig. 6 Estimation of dynamic time windows (bold values represent the time reduction percentage compared to the original time window, which is 1 h)

Table 9 Confusion matrix of classification results after introducing dynamic time window

	T.W	BC	BT	EM	VG	TD	BG
BC	432	**100%**	0	0	0	0	0
BT	38	0	**100%**	0	0	0	0
EM	411	0	0	**100%**	0	0	0
VG	144	0	0	0	**97.22%**	0	2.78%
TD	2130	0	0	0	0	**100%**	0
FPR	–	0	0	0	0	0	–

T.W: total number of time windows. BC: BitComet; BT: BitTorrent; EM: eMule; VG: Vagaa; TD: Thunder; BG: background. FPR: false positive rate. Bold values represent the true positive rate (TPR) of the corresponding application

By contrast, our approach is able to work well with complex host traffic. Neither statistical nor behavioral feature is required in our approach. What we have to do is just recognizing specific flows through their packets sum and size. These flows, i.e., **CF**, are strong evidence of the existence of corresponding P2P applications. Therefore, our classification methodology would not be affected by the impact of unexpected traffic. We estimate the performance of our approach in a complex host traffic context using mixed traces, collected from hosts where P2P file-sharing applications are run with other high bandwidth-consumers for 1 h. For instance, host H_1 runs BC together with BT, host H_2 simultaneously runs both BC and TD, while host H_3 watches videos through a web browser while waiting for P2P files to download by EM. The results are illustrated in Table 10. The percentages reported in the table represent the identification accuracy rates of corresponding P2P applications in all of their time windows. Since we employed a dynamic time window mechanism, the sums of time windows of every P2P application vary from each other, although they all run for 1 h. The time windows of different P2P applications on the same host are parallel and overlapped. Therefore, we are able to identify two or more different P2P applications simultaneously running on the same host. As shown in the table, all targets are completely identified in all of their dynamic time windows, even the applications running on the same host and based on the same P2P protocol.

Table 10 Confusion matrix of classification results in complex network context

Host	App.	T.W	BC	BT	EM	VG	TD
H_1	BC	169	100%	0	0	0	0
	BT	63	0	100%	0	0	0
H_2	BC	271	100%	0	0	0	0
	TD	80	0	0	0	0	100%
H_3	EM	116	0	0	100%	0	0

App.: application. T.W: total number of time windows. BC: BitComet; BT: BitTorrent; EM: eMule; VG: Vagaa; TD: Thunder

6 Conclusions

In this paper, we presented a fine-grained host-based P2P traffic classification approach. This study aims to locate all P2P hosts within a monitored network and identify the types of P2P applications they are running. The novelty of our approach lies in two aspects. First, we classify P2P applications by simply counting some special flows, i.e., clustering flows. No additional complicated information except several generic properties of flows is needed. Second, our approach can work well with hosts in a complex network context, while most existing host-based approaches cannot deal with this situation.

The performance of our approach has been evaluated with real-world traffic. The experimental results show that we can classify P2P file-sharing applications with a TPR higher than 97.22% and a FPR lower than 2.78%. In addition, our approach is capable of classifying P2P applications even when their hosts are simultaneously running other high bandwidth-consuming applications.

Our approach provides a promising reference for traffic classification in a high-speed network, because of its simplicity and flexibility. More types of P2P applications will be considered in our future work, including P2P-TV, VoIP, etc.

References

Auld, T., Moore, A.W., Gull, S.F., 2007. Bayesian neural networks for Internet traffic classification. *IEEE Trans. Neur. Netw.*, **18**(1):223-239. [doi:10.1109/TNN.2006.883010]

Ban, T., Guo, S., Eto, M., *et al.*, 2012. A study on cost-effective P2P traffic classification. Proc. Int. Joint Conf. on Neural Networks, p.1-7. [doi:10.1109/IJCNN.2012.6252672]

Basher, N., Mahanti, A., Mahanti, A., *et al.*, 2008. A comparative analysis of web and peer-to-peer traffic. Proc. 17th Int. Conf. on World Wide Web, p.287-296. [doi:10.1145/1367497.1367537]

Bermolen, P., Mellia, M., Meo, M., *et al.*, 2011. Abacus: accurate behavioral classification of P2P-TV traffic. *Comput. Netw.*, **55**(6):1394-1411. [doi:10.1016/j.comnet.2010.12.004]

Chen, J.B., 2011. Fuzzy based approach for P2P file sharing detection. *J. Internet Technol.*, **12**(6):921-930.

Dainotti, A., Pescapè, A., Claffy, K.C., 2012. Issues and future directions in traffic classification. *IEEE Network*, **26**(1):35-40. [doi:10.1109/MNET.2012.6135854]

Dhamankar, R., King, R., 2007. Protocol Identification via Statistical Analysis (PISA). White Paper, Tipping Point.

Este, A., Gringoli, F., Salgarelli, L., 2009. On the stability of the information carried by traffic flow features at the packet level. *ACM SIGCOMM Comput. Commun. Rev.*, **39**(3):13-18. [doi:10.1145/1568613.1568616]

Finamore, A., Mellia, M., Meo, M., *et al.*, 2010. KISS: stochastic packet inspection classifier for UDP traffic. *IEEE/ACM Trans. Netw.*, **18**(5):1505-1515. [doi:10.1109/TNET.2010.2044046]

Gallagher, B., Iliofotou, M., Eliassi-Rad, T., *et al.*, 2010. Link homophily in the application layer and its usage in traffic classification. Proc. IEEE INFOCOM, p.1-5. [doi:10.1109/INFCOM.2010.5462239]

Gomes, J.V., Inácio, P.R.M., Pereira, M., *et al.*, 2013. Detection and classification of peer-to-peer traffic: a survey. *ACM Comput. Surv.*, **45**(3), Article 30. [doi:10.1145/2480741.2480747]

He, J., Yang, Y., Qiao, Y., *et al.*, 2013. Accurate classification of P2P traffic by clustering flows. *China Commun.*, **10**(11):42-51. [doi:10.1109/CC.2013.6674209]

Huang, N.F., Jai, G.Y., Chao, H.C., 2008. Early identifying application traffic with application characteristics. Proc. IEEE Int. Conf. on Communications, p.5788-5792. [doi:10.1109/ICC.2008.1083]

Hullár, B., Laki, S., Gyorgy, A., 2011. Early identification of peer-to-peer traffic. Proc. IEEE Int. Conf. on Communications, p.1-6. [doi:10.1109/icc.2011.5963023]

Hurley, J., Garcia-Palacios, E., Sezer, S., 2011. Host-based P2P flow identification and use in real-time. *ACM Trans. Web*, **5**(2), Article 7. [doi:10.1145/1961659.1961661]

Iliofotou, M., Kim, H., Faloutsos, M., *et al.*, 2011. Graption: a graph-based P2P traffic classification framework for the Internet backbone. *Comput. Netw.*, **55**(8):1909-1920. [doi:10.1016/j.comnet.2011.01.020]

Karagiannis, T., Papagiannaki, K., Faloutsos, M., 2005. BLINC: multilevel traffic classification in the dark. *ACM SIGCOMM Comput. Commun. Rev.*, **35**(4):229-240. [doi:10.1145/1090191.1080119]

Moore, A., Zuev, D., Crogan, M., 2005. Discriminators for Use in Flow-Based Classification. Technical Report, University of London, UK.

Nguyen, T.T.T., Armitage, G., 2008. Clustering to assist supervised machine learning for real-time IP traffic classification. Proc. IEEE Int. Conf. on Communications, p.5857-5862. [doi:10.1109/ICC.2008.1095]

Ohzahata, S., Hagiwara, Y., Terada, M., *et al.*, 2005. A traffic identification method and evaluations for a pure P2P application. Proc. 6th Int. Workshop on Passive and Active Network Measurement, p.55-68. [doi:10.1007/978-3-540-31966-5_5]

Sandvine, 2014. Global Internet Phenomena Report 1H 2014. Technical Report. Sandvine Incorporated ULC, Waterloo, Ontario, Canada.

Tabatabaei, T.S., Adel, M., Karray, F., *et al.*, 2012. Machine learning-based classification of encrypted Internet traffic. Proc. 8th Int. Conf. on Machine Learning and Data Mining in Pattern Recognition, p.578-592. [doi:10.1007/978-3-642-31537-4_45]

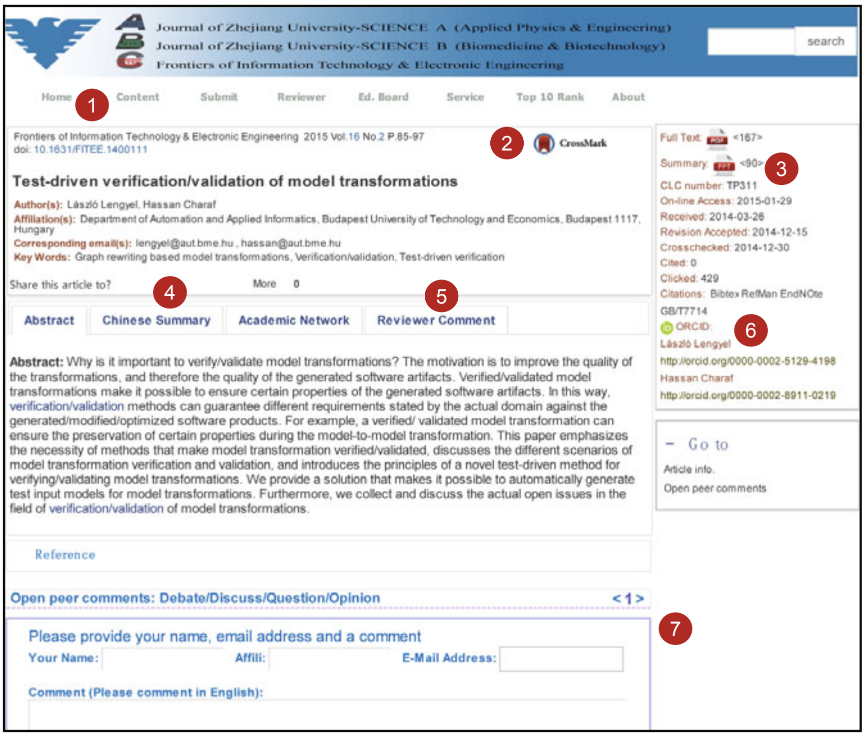
Valenti, S., Rossi, D., 2011. Identifying key features for P2P traffic classification. Proc. IEEE Int. Conf. on Communications, p.1-6. [doi:10.1109/icc.2011.5963018]

Yang, D., Zhang, Y., Zhang, H., *et al.*, 2009. Multi-factors oriented study of P2P Churn. *Int. J. Commun. Syst.*, **22**(9):1089-1103. [doi:10.1002/dac.1001]

Zhang, T., Ramakrishnan, R., Livny, M., 1996. BIRCH: an efficient data clustering method for very large databases. *ACM SIGMOD Rec.*, **25**(2):103-114. [doi:10.1145/235968.233324]

Highlights of *JZUS (A/B)* & *FITEE* New Website in 2015

(http://www.zju.edu.cn/jzus/)



Journal of Zhejiang University-SCIENCE A (Applied Physics & Engineering)
Journal of Zhejiang University-SCIENCE B (Biomedicine & Biotechnology)
Frontiers of Information Technology & Electronic Engineering

Frontiers of Information Technology & Electronic Engineering 2015 Vol.16 No.2 P.85-97
doi: 10.1631/FITEE.1400111

Test-driven verification/validation of model transformations

Author(s): László Lengyel, Hassan Charaf
Affiliation(s): Department of Automation and Applied Informatics, Budapest University of Technology and Economics, Budapest 1117, Hungary
Corresponding email(s): lengyel@aut.bme.hu , hassan@aut.bme.hu
Key Words: Graph rewriting based model transformations, Verification/validation, Test-driven verification

Share this article to? More 0

Abstract: Why is it important to verify/validate model transformations? The motivation is to improve the quality of the transformations, and therefore the quality of the generated software artifacts. Verified/validated model transformations make it possible to ensure certain properties of the generated software artifacts. In this way, verification/validation methods can guarantee different requirements stated by the actual domain against the generated/modified/optimized software products. For example, a verified/ validated model transformation can ensure the preservation of certain properties during the model-to-model transformation. This paper emphasizes the necessity of methods that make model transformation verified/validated, discusses the different scenarios of model transformation verification and validation, and introduces the principles of a novel test-driven method for verifying/validating model transformations. We provide a solution that makes it possible to automatically generate test input models for model transformations. Furthermore, we collect and discuss the actual open issues in the field of verification/validation of model transformations.

Full Text: <167>
Summary: <90>
CLC number: TP311
On-line Access: 2015-01-29
Received: 2014-03-26
Revision Accepted: 2014-12-15
Crosschecked: 2014-12-30
Cited: 0
Clicked: 429
Citations: Bibtex RefMan EndNOte
GB/T7714
ORCID:
László Lengyel
http://orcid.org/0000-0002-5129-4198
Hassan Charaf
http://orcid.org/0000-0002-8911-0219

Open peer comments: Debate/Discuss/Question/Opinion <1>

Please provide your name, email address and a comment
Your Name: Affili: E-Mail Address:
Comment (Please comment in English):

1 New title for *JZUS-C* since 2015 (FITEE)

2 CrossMark: tracking content changes

3 English summary (PPT)

4 Chinese summary

5 Selected comments from peer reviewers (before publication)

6 ORCID: connecting research and researchers

7 Open peer comments (after publication)