Future Generation Computer Systems 88 (2018) 0–13

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

# Extending labeled mobile network traffic data by three levels traffic identification fusion

Zhen Liu [a], Ruoyu Wang [b,c,*], Deyu Tang [a]

[a] College of Medical Information Engineering, Guangdong Pharmaceutical University, Guangzhou, China
[b] Information Network Engineering and Research Center, South China University of Technology, Guangzhou, China
[c] Communication & Computer Network Lab of Guangdong, Guangzhou, China

## HIGHLIGHTS

- A method named ELD is devised to automatically extend the labeled mobile traffic data.
- Payload distribution inspection (PDI) is proposed to identify the label of unknown traffic on packet payload level.
- Bi-direction payload distribution pattern (BPDP) is presented to reflect the byte distribution pattern of a flow.
- ELD outperforms existing works (nDPI and Libprotoident) on labeling mobile network traffic.

## ARTICLE INFO

*Article history:*
Received 11 May 2017
Received in revised form 23 November 2017
Accepted 29 May 2018
Available online 15 June 2018

*Keywords:*
Mobile network traffic
Labeled data
Traffic identification
Packet payload
Machine learning

## ABSTRACT

Mobile traffic classification is critically important for the decision-making of network management such as traffic shaping and traffic pricing. Labeled traffic data are the requisite of classification performance evaluation. However, existing works mostly acquired labeled traffic on a simulation environment such as individually running a specific app on mobile devices to collect its traffic. This way is slow and not scalable. This paper devises a scheme to automatically link the ground truth to mobile traffic. A set of labeled traffic data are firstly collected by our previously presented *mobilegt* (a system to collect **mobile** traffic and build the **g**round **t**ruth) on the monitored mobile devices. But these traffic are limited to the monitored nodes. Therefore, we present a method named ELD (**E**xtending **L**abeled **D**ata) to identify the label of newly unknown mobile traffic, so as to extend the labeled mobile traffic data. ELD proceeds traffic identification into packet header, packet payload and flow statistic levels. The three levels' traffic identification tasks are implemented by ServerTag, payload distribution inspection and Random Forest respectively. ELD is able to identify the mobile traffic with encrypted payload. The cross validation results show that ELD achieves 99% flow accuracy and 95.4% byte accuracy on average when the flow and byte completeness are respectively 86.5% and 65.5%. The results also prove that ELD outperforms existing works, nDPI and Libprotoident, on labeling mobile network traffic.

© 2018 Elsevier B.V. All rights reserved.

## 1. Introduction

Recent years have witnessed the increased popularity of mobile apps such as WhatsApp and WeChat etc. Mobile apps bring us so many convenience, such as video chatting and web searching at anywhere and anytime. With the popular usage of mobile apps, a large amount of mixed traffic are generated every day, leading to difficult management of mobile network. These mixed traffic also contain much valuable information. Mobile traffic classification is helpful for business and mobile network management. For example, it can help app providers understand which mobile apps are popular [1]; it can also help network managers know which apps consume much more network bandwidth, so as to help managers make a decision on suitably allocating mobile network bandwidth. However, a key task of classification algorithm research is to collect labeled mobile traffic data.

Methods on collecting labeled traffic data generally fall into the following three categories. First one is to collect raw traffic traces on a lived cellular network [2]. And the raw traffic could be further labeled by DPI-based and port-based techniques [3]. DPI (deep packet inspection) methods generally rely on the signature repository for matching protocols [4–6]. However, traditional signatures (e.g. regular expression of L7-filter [4]) may be not suitable for mobile traffic. And DPI based techniques cannot handle encrypted traffic. Second one is to capture the traffic of an app each time by

* Corresponding author at: Information Network Engineering and Research Center, South China University of Technology, Guangzhou, China.
*E-mail address:* rywang@scut.edu.cn (R. Wang).

https://doi.org/10.1016/j.future.2018.05.079
0167-739X/© 2018 Elsevier B.V. All rights reserved.

manually running it on mobile devices [7]. However, such traffic are not from real usage context and cannot be used to analyze the characteristics of mixed traffic since only one app is running when collecting the traffic. And this way is slow and not scalable. Third one is to deploy a special app on monitored mobile devices to collect the socket information that records the association between user apps and active sessions [8]. The mobile traffic of monitored devices are collected and further labeled according to the socket information. But the collected traffic are limited to monitored devices.

To overcome the above problems, this paper presents a method to extend the labeled data at the base of the initial labeled traffic data collected by the third way mentioned above. The main contributions of this paper are as follows.

(1) *Mobilegt* [8] was deployed to acquire initial labeled traffic data. It retrieves the app name for each flow from the socket information recorded on mobile devices. *Mobilegt* can obtain 100% accuracy on labeling mobile traffic. But the labeled mobile traffic are limited to monitored devices.

(2) To extend the labeled traffic data, a cascade method named ELD (Extending Labeled Data) is presented to identify the traffic class (label) of unknown traffic. ELD proceeds traffic identification into three levels, respectively, packet header, packet payload and flow statistic. The three levels' traffic identification is respectively implemented by ServerTag, payload distribution inspection (PDI) and machine learning technique (Random Forest [9] used here). At learning stage, on the initial labeled traffic data, the association between traffic class and server IP address is recorded for ServerTag; the payload distribution patterns of each traffic class are automatically extracted for PDI; and flow statistical feature characterized samples are used to train a classification model. At traffic identification stage, the flows with recorded servers are firstly predicted by ServerTag, and then the flows closing to extracted payload distribution patterns are identified by PDI. At last all rest of unknown traffic are fed into the classification model. The flow with high score will be labeled as the prediction class.

(3) Experiments are carried out on 30 mobile traffic datasets, which include popularly used mobile apps. ELD is compared against existing works on labeling mobile traffic, i.e., nDPI [10] and Libprotoident [11]. Results show that ELD outperforms others in terms of flow accuracy and byte accuracy.

The rest of this paper is organized as follows. Section 2 overviews related work on mobile traffic classification and mobile traffic collection. Section 3 devises a scheme to acquire labeled mobile traffic data and presents a method named ELD to extend the labeled mobile traffic data. Section 4 describes experimental data and performance evaluation metrics. Section 5 carries out experiments to compare ELD against existing works and discusses experimental results. Section 6 concludes this paper.

## 2. Related work

### 2.1. Mobile traffic classification methods

Existing mobile traffic classification methods could be summarized into following three levels: packet header, packet payload and flow statistic.

On packet header level, the port number was used to discriminate network traffic at the early stage of network traffic classification, but port number based techniques have been ineffective as the popular usage of dynamic port numbers [12]. In mobile network, HTTP traffic classification attracts a great attention as HTTP is popularly used by mobile apps for communication [13]. The IP address in packet header has been used to classify HTTP traffic. Recently, HTTPTag [14] was presented to classify HTTP traffic into 280 services based on analyzing the association between the service and the IP address (in a packet header) assigned to the server providing the content. HTTPTag enlightened us to introduce the packet header level method into our method.

On packet payload level, existing works [13,15,16] focused on extracting different fields of HTTP payload header to identify mobile apps. Recently, Ranjan et al. [13] presented a method named AMPLES to address mobile application identification problem. It extracted HST (host-names) and URI (uniform resource identifier) fields to build the association between app and these fields. The association will be used to identify the apps of unknown HTTP traffic. Han et al. [17] presented maximum entropy based mobile traffic classification method, in which the entropy of payload blocks were extracted as payload features fed into machine learning technique. The experimental data mainly include web browsing, video streaming, VoIP and FTP traffic. But payload based methods cannot handle encrypted traffic.

On flow statistic level, a set of flow statistical features were extracted to discriminate mobile traffic, such as flow length, flow size and inter arrival time etc. [6]. Satoh et al. [18] researched the sub flow based traffic classification method, i.e., the flow statistical features were extracted from sub flows rather than full flows, so as to classify mobile traffic at early stage. They presented a method to select sub-flow according to application behavior before training a classifier. The experiment data contain HTTP, SMTP and IMAP traffic. Mongkolluksamee et al. [19] presented a mobile traffic classification method based on packet distribution and communication behavior features. The behavior feature set is composed by the number of flows of one node in each direction, the number of nodes with one flow etc. The experiment data contain five popularly used traffic, i.e., Facebook, Line, Skype, Web and YouTube. However, the flow statistical features are easily influenced by network environment, and are less stable than payload signatures.

All mobile traffic classification researches require the labeled traffic data for classification performance evaluation. The related works on collecting labeled mobile traffic will be overviewed in next section.

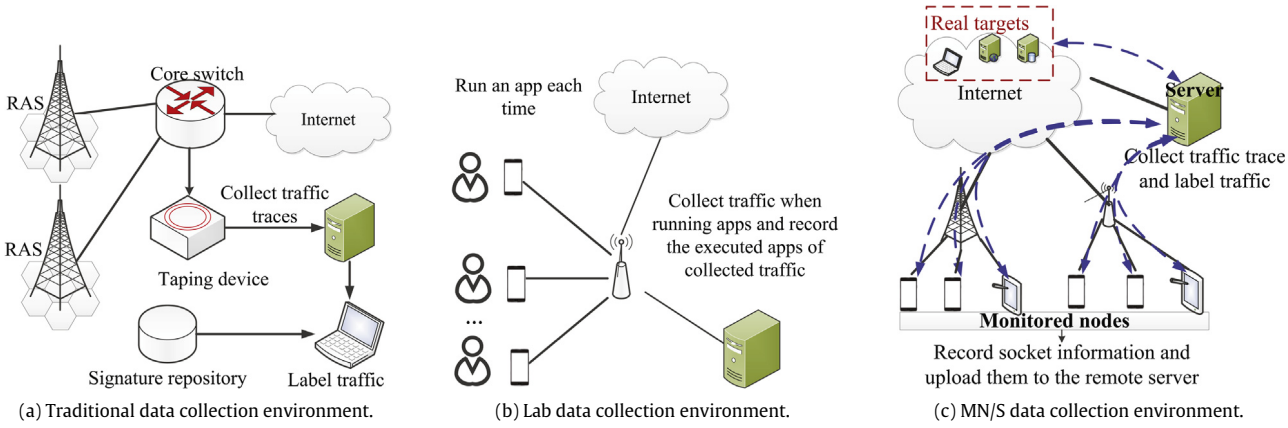
### 2.2. Labeled mobile traffic collection methods

Labeled mobile traffic collection generally proceeds into two steps: (1) collecting raw traffic, (2) building the ground truth for the collected traffic by specific methods.

In the traditional data collection environment, the probes could be deployed at different locations [2] to capture traffic traces, such as the core switch in Fig. 1(a). The traffic could be further labeled by inspecting packet payload at the aid of signature repository, such as L7-filter [4], nDPI [10] and Libprotoident [11] etc. The signature repository is generally built for matching the protocols of traffic traces. But this way requires cellular network manager authority, and existing protocol signatures may be ineffective on mobile traffic as HTTP protocol is popularly used by mobile apps [5].

In the lab data collection environment shown in Fig. 1(b), the traffic of an app are collected each time [1,13,17,19]. In this way, an active app is individually executed in a certain period of time. The collected traffic during this period could be labeled as the executed app, which does not rely on predefined payload signature. For example, Ranjan et al. [13] captured the traffic of 400k apps in an emulator environment. They executed an app thrice, and each execution instance lasted 5 min. When executing an app, automated random clicks were arbitrarily performed on the GUI of the app. But this way is slow and not scalable.

To handle this problem, we presented *mobilegt* [8] to collect mobile traffic and to build the ground truth automatically in MN/S (Monitored Nodes and Server) environment (shown in Fig. 1(c)). An app (named *mgtClient*) is installed on volunteers' mobile devices to collect the socket information of active TCP/UDP sessions. Volunteers use apps as usual, and the generated traffic will be routed




(a) Traditional data collection environment. (b) Lab data collection environment. (c) MN/S data collection environment.

**Fig. 1.** Mobile traffic data collection environment.

to a server and be labeled at the base of the socket information. *Mobilegt* is the improved version of gt [20] which was proposed for traditional network.

A variety of mobile traffic datasets associated with different traffic class sets were collected by existing works [21]. A common characteristic of existing traffic class sets is that they are grouped on popularly used apps [22,23], such as Web browsing, Video streaming, etc. This paper also mainly concerns the popular groups.

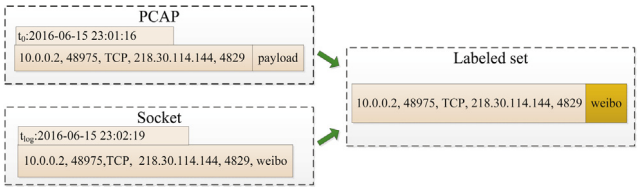



**Fig. 2.** Mobile traffic labeling scheme by *mobilegt*.

## 3. Method on building labeled mobile traffic data

### 3.1. Labeled mobile traffic

The initial labeled mobile traffic data were obtained by running our *mobilegt* [8] system.

#### 3.1.1. Mobilegt

*Mobilegt* was presented to tackle two tasks: collecting mobile traffic and labeling collected traffic. It consists of two components: an android app named *mgtClient* and a server application named *mgtServer*. *MgtClient* is deployed on mobile devices and *mgtServer* is deployed on a remote server. The two tasks of *mobilegt* are detailed as below.

(1) Collecting raw traffic

*Mobilegt* system collects raw traffic traces based on VPN (Virtual Private Network). *MgtClient* firstly starts a VPN service and connects to *mgtServer*. Once the connection is successfully built, *mgtServer* becomes the communication bridge between monitored devices and Internet. This means that all outgoing and incoming traffic of monitored devices will go across the remote server. Therefore, raw traffic traces of monitored devices are captured on the server side using a common tool (e.g. *tShark*) and stored into PCAP (Process Characterization Analysis Package) format.

In order to tag mobile traffic, we require the socket information of these traffic. *MgtClient* starts to retrieve the association between sockets and apps once the connection between *mgtClient* and *mgtServer* is successfully built. A thread samples at a fixed interval the state of active sockets as recorded by the kernel, and assigns an app to each socket based on the association between sockets and apps. The interval is set as 1 s in *mobilegt* according to our empirical results [8]. All socket information will be stored and transferred to the remote server iteratively.

(2) Labeling raw traffic

When labeling raw traffic traces, *mgtServer* firstly extracts the three tuple {destination IP, destination Port, Protocol} of each packet. When observing a packet with timestamp $t_0$, *mgtServer* looks in the socket information with the same three tuple and a log time $t_{log}$ closed to $t_0$ in a time window. If found, the associated app is the label of the packet. For example, in Fig. 2, when labeling a packet with a three tuple {218.30.114.144, 4829, TCP} and $t_0$, we can find a socket that has the same three tuple with $t_{log}$ closing to $t_0$, and then the packet is labeled as *weibo*.

#### 3.1.2. Labeled mobile traffic

To collect mobile traffic data, *mgtClient* (polling interval is 1 s) is installed on volunteers' smartphones and *mgtServer* is deployed on a remote server in MN/S environment. Once volunteers lunch *mgtClient* and connect to *mgtServer*, the data connection is started. Volunteers could access network by WiFi or 3G/4G, and use mobile apps (such as Wechat, Browser etc.) as usual during data connection. The traffic data were collected during June 2016 to October 2016. On some days, volunteers may forget to start *mgtClient* or only connect to *mgtServer* for a while, leading to no traffic or a small amount of traffic volume. We selected the traffic of 30 days on which more available traffic were labeled by *mobilegt*. The labeled results are shown as Fig. 3. The *x*-axis is the index of datasets and *y*-axis is the ratio of labeled flows/packets/bytes or encrypted bytes. For instance, the ratio of labeled flows is calculated by the number of labeled flows divided by the total number of flows on a dataset. On average, *mobilegt* can tag more than 99% of bytes, 98% of packets and 94% of flows. High ratio of labeled bytes suggests that nearly all traffic were labeled by *mobilegt*. But, these traffic are limited to monitored nodes.

To acquire more labeled data, we further present a method to extend labeled mobile traffic data. Payload inspection method is generally used to build the ground truth for network traffic [6,24]. However, it cannot handle encrypted traffic. The amount encrypted traffic is not small with the popular usage of encryption techniques, e.g. about 23% of encrypted packets on our labeled traffic. Therefore, we combine payload inspection, packet header and flow statistic based methods to extend the labeled mobile traffic data.

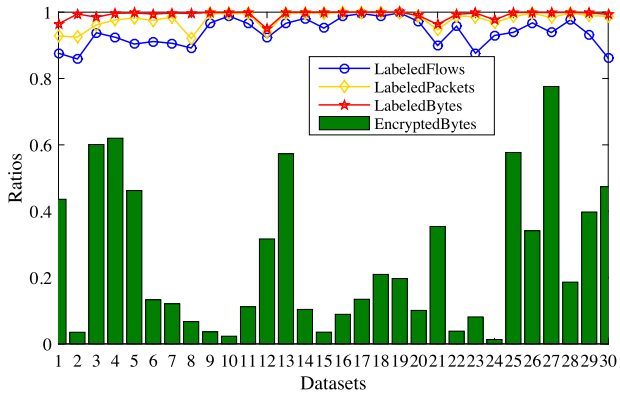



**Fig. 3.** Labeled mobile traffic obtained by *mobilegt*.

### 3.2. Extending labeled mobile traffic

#### 3.2.1. Overview of ELD

ELD identifies mobile traffic from three levels, respectively, packet header, packet payload and flow statistic.

On packet header level, the Server IP address and Port number could be used for traffic identification. However, Port number becomes ineffective as the popular usage of HTTP protocol (80 port number). Enlightened by [14], this paper presents a method named ServerTag. It utilizes Server IP address to identify mobile traffic on packet header level. This method can quickly identify unknown traffic, but it cannot correctly identify the traffic generated by unknown servers and the server hosting multiple apps.

On packet payload level, predefined protocol signatures (e.g. L7-filter) were popularly used to build the ground truth for traditional network traffic [24]. But these signatures may be not suitable for mobile traffic as HTTP protocol is popularly used by mobile apps. To obtain good performance, these predefined protocol signatures are required to be updated periodically due to the dynamic characteristic of mobile traffic. However, the updating work is difficult because it often involves manually extracting discriminative signatures. This paper presents a method named PDI (payload distribution inspection) to automatically extract payload distribution pattern from given packet payloads. The payload level method is stable, but it cannot handle encrypted traffic.

On flow statistic level, machine learning techniques have been utilized to classify the traffic flows characterized by flow statistical features [25] without inspecting packet payloads. The Random Forest performs the best on mobile traffic data according to our experimental results in Section 5.3.3. Flow statistic level method can handle encrypted traffic. But flow statistical features may be drift because of the unstable of network environment or the variation of app versions. This means that the traffic classifier faces concept drift problem [25].

ELD is the composition of ServerTag, PDI and Random Forest by effectively utilizing each method's strengths. And it pursues high accuracy on identification, so as to label newly unknown traffic. The framework of ELD is shown in Fig. 4. It mainly proceeds into the following two steps.

(1) At learning stage, the feature sets for ServerTag, PDI and Random Forest are firstly extracted on initial labeled data, and are used to train three models for mobile traffic identification.

(2) At identification stage, the labels of unknown traffic flows are predicted by ServerTag, PDI and Random Forest in sequence. The flows with high scores will be tagged and added to the newly labeled mobile traffic data.

Using ELD, mobile traffic identification proceeds in a cascade way. The flow chart is shown as Fig. 5. We firstly divide the traffic into HTTP traffic and non-HTTP traffic. On HTTP traffic, ServerTag firstly tags the flows in server host set. And then the rest of traffic are divided into encrypted traffic and unencrypted traffic. On the unencrypted traffic, PDI predicts the label of each unknown flow based on the payload distribution pattern. And all the rest of traffic are fed into the ensemble classifier trained by Random Forest algorithm. The ensemble classifier predicts each unknown flow as the class with the highest vote among individuals. The rest of flows are tagged as unknown. Fig. 5 shows that the encrypted traffic and the traffic without payload could be labeled by ServerTag and Random Forest.

Before applying ELD, the flows are required to be extracted from raw traffic traces. The definition of a flow is given by Definition 2 in Section 3.2.2. Let $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ denote a training set, where $x_i$ refers to the *ith* flow and $y_i$ is the class label of $x_i$. The $y_i$ is from a predefined traffic class set $C = \{c_1, c_2, \ldots, c_m\}$ with $m$ classes.

#### 3.2.2. ServerTag

Similar to the HTTPTag [14], we record the association between server IP address and traffic class (Tclass) on HTTP traffic. In contrast to HTTPTag, we do not need to inspect the URL in HTTP header because we can simply find the app using *mobilegt*.

**Definition 1** (*ServerTag <IP, Tclass>*)**.** ServerTag denotes the association between server IP address and traffic class, $<IP_i, c_i>$. $IP_i$ is a set of server IP addresses hosting service of $c_i$, $IP_i = \{ip_i^{(1)}, ip_i^{(2)}, \ldots, ip_i^{(k)}\}$, $i = 1, \ldots, m$ . The $m$ server-IP-address sets are disjoint.

When classifying an unknown flow $u_{new}$, its server IP address ($ip_{new}$) is extracted and used to predict the traffic class of $u_{new}$ based on the ServerTag of each class $c_i$, $F(u_{new}) = c_i$, if $ip_{new} \in IP_i$.

With the popular usage of third-party servers, a server may provide multiple services. As a results, different traffic classes' flows may come from the same server, leading to overlapping between $m$ server-IP-address sets. Such IP address would not be included in ServerTag. Therefore, the traffic generated by these servers would be ignored by ServerTag. And these traffic would be fed into next stage of ELD. This means that the traffic generated by the server providing multiple services would not be handled by serverTag.

#### 3.2.3. Payload distribution inspection

**(1) Bi-direction payload distribution pattern**

Given the application layer header, we can extract useful signatures (such as GET and POST in HTTP packets) for mobile traffic identification. However, traditional payload inspection methods are often cumbersome and complex, since they involve manually extracting discriminative tokens and regular expressions [4,10]. In contrast to it, this paper presents a method to automatically extract a bi-direction payload distribution pattern (BPDP) from the first $K$ bytes of inspected packets on each flow. On the labeled traffic, we can obtain a set of BPDPs (denoted by $P_i$) for each traffic class $c_i$, $i = 1, 2, \ldots m$ . When identifying an unknown flow $u_{new}$, the BPDP ($P_{new}$) will be extracted and used to predict the traffic class of $u_{new}$ based on the $m$ BPDP sets. That is, $\text{PDI}(u_{new}) = c_i$, if the similarity between $P_i$ and $P_{new}$ is the maximum among $m$ BPDP sets.

**Definition 2** (*Flow*)**.** A flow is a single bidirectional communication between two end nodes characterized by a 5-tuple {*source IP, source Port, destination IP, destination Port, protocol*}.

**Definition 3** (*Inspected Packet*)**.** The inspected packet refers to the first packet with non-zero payload in each direction on a flow.

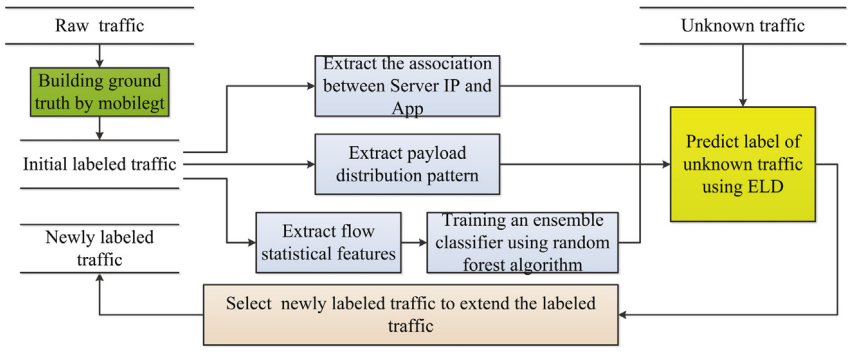



**Fig. 4.** Framework of ELD.

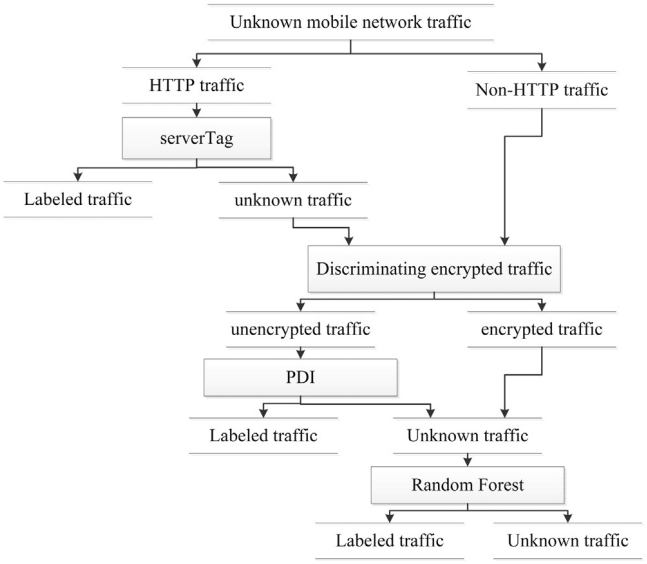



**Fig. 5.** The flow chart of mobile traffic identification using ELD.

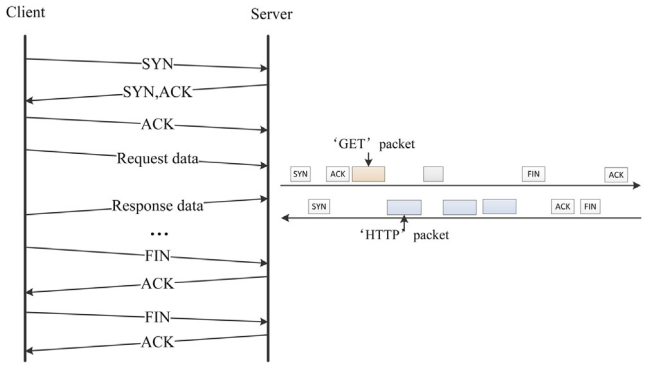



**Fig. 6.** HTTP communication packets.

Taking HTTP communication as an example, in Fig. 6, after three handshake, the client sends data request packet to the server (the header of the payload contains GET), and then the server sends response packet to the client (the header of the payload contains HTTP). The 'GET' and 'HTTP' packets are the inspected packets and will be fed into BPDP extraction. This means that the BPDP is obtained on the first $K$ bytes of in-direction packet and out-direction packet on each flow. We are mainly interested in the application layer header information for each flow. For TCP mobile traffic, our expectation information is transferred at the very start of a flow. For UDP mobile traffic, our expected information is transmitted in every packet at the beginning of the payload [26]. This means that the TCP flow without SYN packet will be ignored by PDI and fed into next stage.

To extract BPDP, $n$-gram [3] is introduced into our method. PDI firstly breaks the $K$ bytes of each inspected packet into $n$-grams, and then records the distribution of each $n$-gram.

**Definition 4** (*n-gram*)**.** An $n$-gram is denoted by the string with $n$ bytes of the first $K$ bytes ($K > n$) from the inspected packet payload. For instance, the first $K$ bytes ("HTTP1.0") of a HTTP packet payload can be represented with the following $n$-grams:

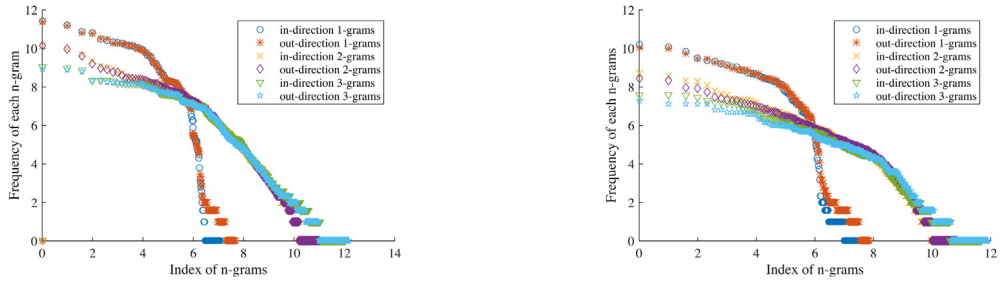
1-gram: H, T, T, P, 1,., 0
2-gram: HT, TT, TP, P1, 1.,.0
3-gram: HTT, TTP, TP1, P1., 1.0.

If the $n$ is too short, the grams cannot reflect the byte distribution, and cannot be used for discriminating different payload distribution patterns. If the $n$ is too long, the number of grams would increase exponentially.

To analyze the suitable value of $n$, Fig. 7 shows the distribution of $n$-grams on two popular mobile apps: Browser and Youku. The value of parameter $n$ is varied on the two apps. The $x$-axis denotes the index of $n$-grams, where $n$-grams are sorted in descending order according to the frequency. And $y$-axis denotes the frequency of each $n$-gram on $x$-axis. The $x$-axis and $y$-axis are in logarithmic scale, so as to clearly distinguish the distribution of different n-grams. Fig. 7 shows that the distribution of $n$-grams is highly skewed. When the value of $n$ is bigger, the value range of an $n$-gram is wider. For example, when $n = 1$, the value range of a 1-gram is $[0, 2^8]$. When $n = 2$, the value range of a 2-gram is $[0, 2^{16}]$. We set the value of $n$ to be 2 in our study, as the distribution becomes highly sparse. There is a small part of values are high frequency and could be used to characterize the traffic class.

**Definition 5** (*BPDP*)**.** After $n$-gram generation, a flow $x$ can be represented by a list of terms $G(x) = \{w_1^{(i)}, w_2^{(i)}, \ldots, w_{n_{ki}}^{(i)}, w_1^{(o)}, w_2^{(o)}, \ldots, w_{n_{ko}}^{(o)}\}$. The $n_{ki}$ and $n_{ko}$ respectively represent the number of terms in in-direction and out-direction packets. The $w_j^{(i)}$ represents the $j$th term on in-direction inspect packet, and $w_j^{(o)}$ represents the $j$th term on out-direction inspect packet. A term of BPDP is composed by the value of a 2-gram and the frequency, i.e., $w = \langle v, r \rangle$, where $v$ is the value of 2-gram and $r$ is its frequency.

Fig. 8 shows an example of the BPDP on a HTTP flow. The first two lines are the 2-grams extracted from the out-direction and in-direction inspected packets respectively. The last line is the BPDP calculated from the bi-direction 2-grams. The algorithm to extract BPDP is shown in Algorithm 1. Firstly, the first $K$ bytes of inspected




(a) Browser. (b) Youku.

**Fig. 7.** The distribution of bi-direction $n$-grams on Browser and Youku traffic.

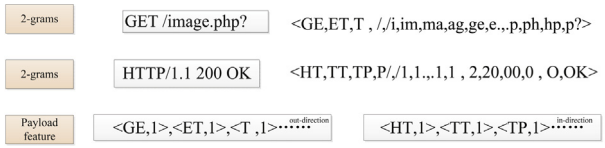



**Fig. 8.** An example of BPDP on a HTTP flow.

packets are split into 2-grams. The $P$ in the line 2 of Algorithm 1 is a set of 2-grams ($P = \{v_1, v_2, \ldots, v_k\}$). And then the frequency of each 2-gram is counted to build BPDP.

**(2) BPDP similarity**

The distance between two BPDPs can be evaluated by the similarity between their 2-grams. We used Jaccard similarity as a metric [27]. The Jaccard similarity between $X$ and $Y$ is denoted by $J(X, Y)$. It is defined as

$$J(X, Y) = \frac{|X \cap Y|}{|X \cup Y|} \tag{1}$$

$J(X, Y)$ approaches one if $X$ and $Y$ are similar and zero otherwise. The $X$ and $Y$ are two different sets of 2-grams on two BPDPs. If two patterns are generated by different apps, each pattern's inspected packets consist of distinct 2-grams and their BPDPs are very different. If two patterns share a lot of 2-grams, they belong to the same app in high probability.

**(3) Traffic flow identification**

Based on the BPDP and the similarity metric, PDI predicts the label of an unknown flow according to the similarity of the BPDP to existing BPDPs extracted from labeled flows. Existing work [28] shows that the bag of flows (BoF) could decrease error variance in network traffic identification. A BoF is composed by some correlated network traffic flows from the same service. All unknown flows in a bag will be predicted to the same class. BoFs could be found by the three tuple heuristic on a real traffic dataset [28], i.e., the flows sharing the same three-tuple {*destination ip, destination port, protocol*} in a certain period of time construct a BoF. This heuristic is based on the assumption that the sessions linked with the same three tuple access the same service. Taking the web service as an example, multiple hosts build TCP connection to a same destination host on 80 port for accessing web service in a period of time, during which all generated flows belong to Web.

An unknown flow set would be partitioned into a list of BoFs after performing the three tuple heuristic. Given an unknown BoF (denoted by $B$), all flows in $B$ will be tagged as the prediction class of $B$. When identifying a BoF, the similarity between $B$ and each existing class is firstly calculated, and then $B$ is predicted to the class with the maximum similarity. The similarity between $B$ and $c_j$ ($j = 1, \ldots, m$) is evaluated by aggregating the BPDP similarity with average operation. It is defined as Eq. (2).

$$D(B, c_j) = \frac{1}{\|B\|} \sum_{u \in B} d(u, c_j) \tag{2}$$

where $d(u, c_j)$ is the similarity between an unknown flow $u$ and $c_j$. It is defined as Eq. (3), where $U$ is a set of 2-grams on $u$, and $P$ is a set of existing 2-grams on labeled flows. The $J(U, P)$ is the Jaccard similarity between $U$ and $P$.

$$d(u, c_j) = \max_{P \in Cj} J(U, P). \tag{3}$$

The prediction function of $B$ is defined as (4). If the similarity between prediction $c_\omega$ and $B$ is higher than a threshold $\alpha$, $B$ will be tagged as $c_\omega$, otherwise *unknown*.

$$f(B) = \begin{cases} c_\omega & D(B, c_\omega) > \alpha \\ unknown & D(B, c_\omega) \leq \alpha \end{cases} \tag{4}$$

where $\omega = \arg\max_j D(B, c_j)$.

Mobile traffic identification algorithm of PDI is shown as Algorithm 2. On the labeled data, we combine the 2-grams of a BoF, i.e., the frequency of the same 2-gram is summed up among all 2-grams extracted from flows in a BoF (line 1). And the 2-grams with high frequency are returned. Each returned $P$ is a set of 2-grams on a BoF of labeled data. And the $M$ is the number of BoFs. On each unknown flow $u_i$ in $B$ , we extract the BPDP from $u_i$ (line 3), and acquire the similarity between $u_i$ and each class based on Eq. (3) (lines 4–8). And then, we calculate the similarity between $B$ and each traffic class according to Eq. (2) (lines 10–12). Finally, we predict the class for $B$ according to Eq. (4) (lines 13–18).

### 3.2.4. Random forest

PDI cannot identify all unknown flows, such as encrypted flows. A machine learning algorithm (Random Forest) is used in the third stage of ELD. It is based on flow statistical features. Fourteen popular features are applied here [6], respectively, *source port, destination port, protocol, number of packets, number of bytes, packet size (minimum, maximum, mean, standard deviation), inter arrival time (minimum, maximum, mean, standard deviation)* and *flow duration*.

Existing research [25] showed that decision tree performs well on network traffic identification. Our experiments in Section 5.3.3 shows that Random Forest performs the best among multiple machine learning techniques. Random forest [9] is an ensemble learning method. Each individual classifier is a decision tree. And each tree is grown as follows. (1) The algorithm randomly samples $N$ (the size of training set) flows from the training set $S$ with replacement. These sampled flows will be used for growing the tree. (2) It selects $l$ flow features from $L$ ($L$ is the total number of flow features on $S$, and $l \ll L$), and choose the best split feature to split the node.

**Algorithm 1** BPDP extraction

**Input**: a flow $x_j$

**Output**: BPDP $\{G^{(i)}(x_j), G^{(o)}(x_j)\}$

if $x_j$ is a TCP flow with SYN packet or $x_j$ is a UDP flow
  $\{P^{(i)}(x_j), P^{(o)}(x_j)\}$= Grams($x_j$); // breaks the first $K$ bytes of bi-direction inspected packets into 2-grams
  $G^{(i)}(x_j)$ = Count($P^{(i)}(x_j)$); //Counting the frequency of each 2-gram on in-direction inspected packet
  $G^{(o)}(x_j)$ = Count($P^{(o)}(x_j)$); //Counting the frequency of each 2-gram on out-direction inspected packet
else
  return null;
end

**Algorithm 2** Identifying an unknown BoF by PDI

**Input:** an unknown BoF, $B=\{u_1,\ldots, u_k\}$, BPDP on labeled data $\{G^{(i)}(x_1), G^{(i)}(x_2),\ldots, G^{(i)}(x_N), G^{(o)}(x_1), G^{(o)}(x_2),\ldots, G^{(o)}(x_N)\}$, $N$ is the number of labeled flows.

**Output:** the traffic class $f(B)$

$\{P^{(i)}{}_1,\ldots, P^{(i)}{}_M, P^{(o)}{}_1,\ldots, P^{(o)}{}_M\}$=Combine($G^{(i)}(x_1),\ldots,G^{(i)}(x_N), G^{(o)}(x_1),\ldots,G^{(o)}(x_N)$) // $M<N$
for $i$ = 1 to $k$
  $\{G^{(i)}(u_i), G^{(o)}(u_i)\}$ = BPDP($u_i$); //BPDP is implemented by Algorithm 1
  for $j$ = 1 to $m$
    $d^{(i)}(u_i, c_j) = \max_{P_t^{(i)} \in C_j}(J(U^{(i)}, P_t^{(i)}))$; //$U^{(i)}$ is the 2-gram set of $G^{(i)}(u_i)$, $P_t^{(i)}$ is the in-direction 2-gram set of the $t$th BoF from $c_j$.
    $d^{(o)}(u_i, c_j) = \max_{P_t^{(o)} \in C_j}(J(U^{(o)}, P_t^{(o)}))$;
    $d(u_i, c_j)=avg(d^{(i)}(u_i, c_j), d^{(o)}(u_i, c_j))$;
  end
end
for $j$=1 to $m$
  $D(B,c_j)=\frac{1}{\|B\|}\sum_{u_i\in B} d(u_i,c_j)$;
end
  $\omega=\arg\max_j D(B,c_j)$;
if($D(B, c_\omega) > \alpha$)
  $f(B)= c_\omega$;
else
  $f(B)$=*unknown*;
end

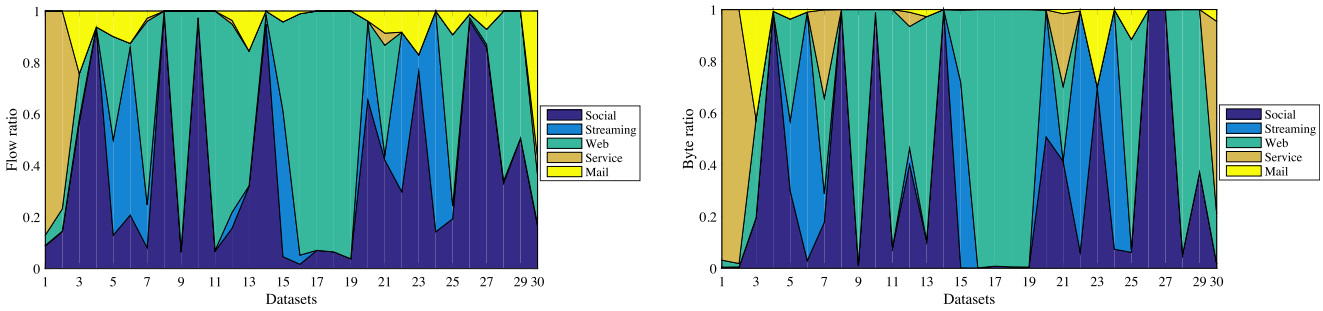
An ensemble classifier would be trained by Random Forest in this stage. When classifying an unknown flow, if the poster probability is higher than a threshold $\beta$, the flow will be tagged as the predication class, otherwise as *unknown*.

## 4. Experimental data and performance evaluation metrics

### 4.1. Mobile traffic data

Our experimental mobile traffic were collected through deploying the *mgtClient* on volunteers' smartphones and the *mgtServer* on a remote server. Volunteers used apps as usual, such as WeChat, Weibo etc. The socket information would be collected on volunteer's smartphone while running *mgtClient*. And the mobile traffic traces of the monitored nodes would be captured and labeled by *mgtServer*. Six volunteers agreed to use *mgtClient* to share their data during June 2016 to October 2016. Among those collected traffic data, we firstly preprocessed them. The usage of *mgtClient* is not a mandatory requirement. Hence, on some days, there is no data, or the data are mostly the background traffic. These data were removed after preprocessing. Finally, 30 datasets from 30 days were utilized to build our experimental data, in which each data set includes 1 to 3 monitored nodes' traffic. These data have been shared in public [29]. And we will collect more traffic data to be shared in public in future when more volunteers would like to share their data. These traffic data will be used to evaluate the performance of our presented ELD. According to our previous research [21], we further group the mobile apps into the following 6 traffic classes (shown in Table 1) on captured traffic data.

The flow and byte distributions on the 30 datasets are shown as Fig. 9. Web and Social have a lot of flows, as the popular usages of Browser, Weibo and WeChat etc. On the aspect of bytes, Web and Social account more bytes than others. On some days, Streaming owns a lot of bytes generated by video streaming apps. On average, Social and Web respectively account 15% and 20% of flows, but account 31% and 38% of bytes respectively. It is well known that HTTP protocol is popularly used by mobile apps. On our labeled traffic, there are about 94% of flows and 66% of bytes generated by HTTP based apps. These traffic are shown in Fig. 10.




(a) Flow ratio. (b) Byte ratio.

**Fig. 9.** Percent of flows and bytes of each traffic class.

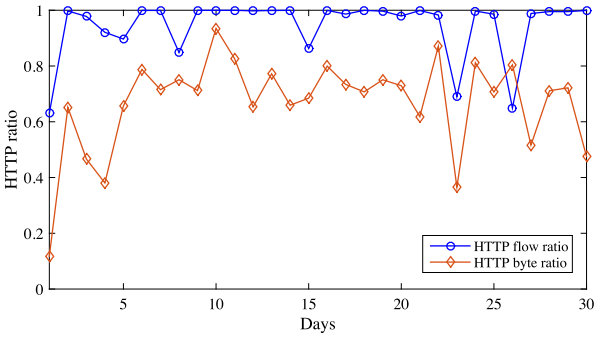



**Fig. 10.** Flow ratio and byte ratio of HTTP traffic.

**Table 1**
Mobile traffic classes.

| Mobile traffic classes | Mobile apps |
|---|---|
| Social | WeChat, Weibo, Facebook, WhatsApp |
| Streaming | Youku, Mi Video |
| Web | Browser, AppStore, VipShop |
| Service | Downloads |
| Mail | Yahoo mail, QQ mail, Gmail |

### 4.2. Performance evaluation metrics and parameter setting

Mobile traffic identification performance will be evaluated from *flow accuracy*, *byte accuracy*, *flow completeness*, *byte completeness*, *flow precision* and *byte precision*.

*Flow* (or *byte*) *accuracy* refers to the ratio between the number of correctly classified flows (or bytes) and the number of classified flows (or bytes) on a traffic dataset.

*Flow* (or *byte*) *completeness* refers to the ratio between the number of classified flows (or bytes) and the total number of flows (or bytes) on a traffic dataset.

*Flow* (or *byte*) *precision* of a class is the ratio between the correctly classified flows (or bytes) in the class and the flows (or bytes) being classified as the class on a traffic dataset.

ELD is implemented by ServerTag, PDI and Random Forest in sequence. In PDI, there are two parameters to be set, (1) the first $K$ bytes of inspected packets is set to be 128, (2) the threshold $\alpha$ for labeling mobile traffic is set to be 0.6. The Random Forest is implemented by Weka toolkit and the default parameters are used. When using Random Forest, the threshold $\beta$ for labeling mobile traffic is set to be 0.9. The parameter discussion will be carried out in Section 5.3.3.

## 5. Experimental results

### 5.1. Payload distribution visualization

In ELD, the PDI predicts the label of mobile traffic at packet payload level. This section visualizes the out-direction payload distribution of three popular used apps on our traffic data: Browser, WeChat and Youku. The in-direction payload distribution has similar results.

We randomly sample 300 flows on training and testing sets for each app. And the visualization results on these flows are shown in Fig. 11. The payload data are broken down into 2-grams, and the value range of each 2-gram is $[0, 2^{16}]$. They are shown in 256 $*$ 256 grid. The value of each point is $(i * 255 + j)$, where $i$ and $j$ are respectively the row and column index. The frequency ratio of each value $v_i$ is counted as Eq. (5), where $r_i$ is the frequency of $v_i$. If the FR of $v_i$ is larger than 0.1, $v_i$ would be shown in Fig. 11.

$$FR(v_i) = \frac{r_i}{\sum_{i=0,\ldots,2^{16}} r_i} \tag{5}$$

Fig. 11(a) to (c) show that the three apps have different payload distribution pattern. And the 2-grams mainly concentrate on the left top area. To further analyze the payload distribution, we zoom in the left top area as following figures. In each figure of Fig. 11(d) to (f), the blue point refers to the payload distribution of an app on training set, and the red one is the dissimilarity between the training and testing sets. In each figure of Fig. 11(g) to (i), the blue point refers to the payload distribution of an app on training set, and the red one is the dissimilarity between an app and another app on training set. The payload distribution of an app on training is similar to that on testing data. Even though the points mainly concentrate on the left top area in each figure, we found that there are a lot of difference between two apps according the results in Fig. 11(g) to (i). This implies that BPDP benefits to discriminate the traffic from different apps.

### 5.2. Performance evaluation results of ELD

#### 5.2.1. Overall identification performance

To avoid the influence of bias and artifacts, the performance of ELD is evaluated by the 10 fold cross validation. Every three datasets are used as testing sets and the rest of datasets are used as training sets among the 30 datasets mentioned in Section 4.1. And ELD is compared against existing works, i.e., nDPI and Libprotoident. The results of the 10 independent experiments are shown in Fig. 12. The $x$-axis is the index of testing set (or experiment) and $y$-axis refers to the result of each performance metric.

Carela-Español et al. [30] carried out a variety of experiments to compare the performance of six tools (i.e., PACE, OpenDPI, L7-filter, nDPI, Libprotoident, and NBAR) used to build ground truth for network traffic. The results in [30] show that nDPI and Libprotoident perform better than others among the open-source tools, and

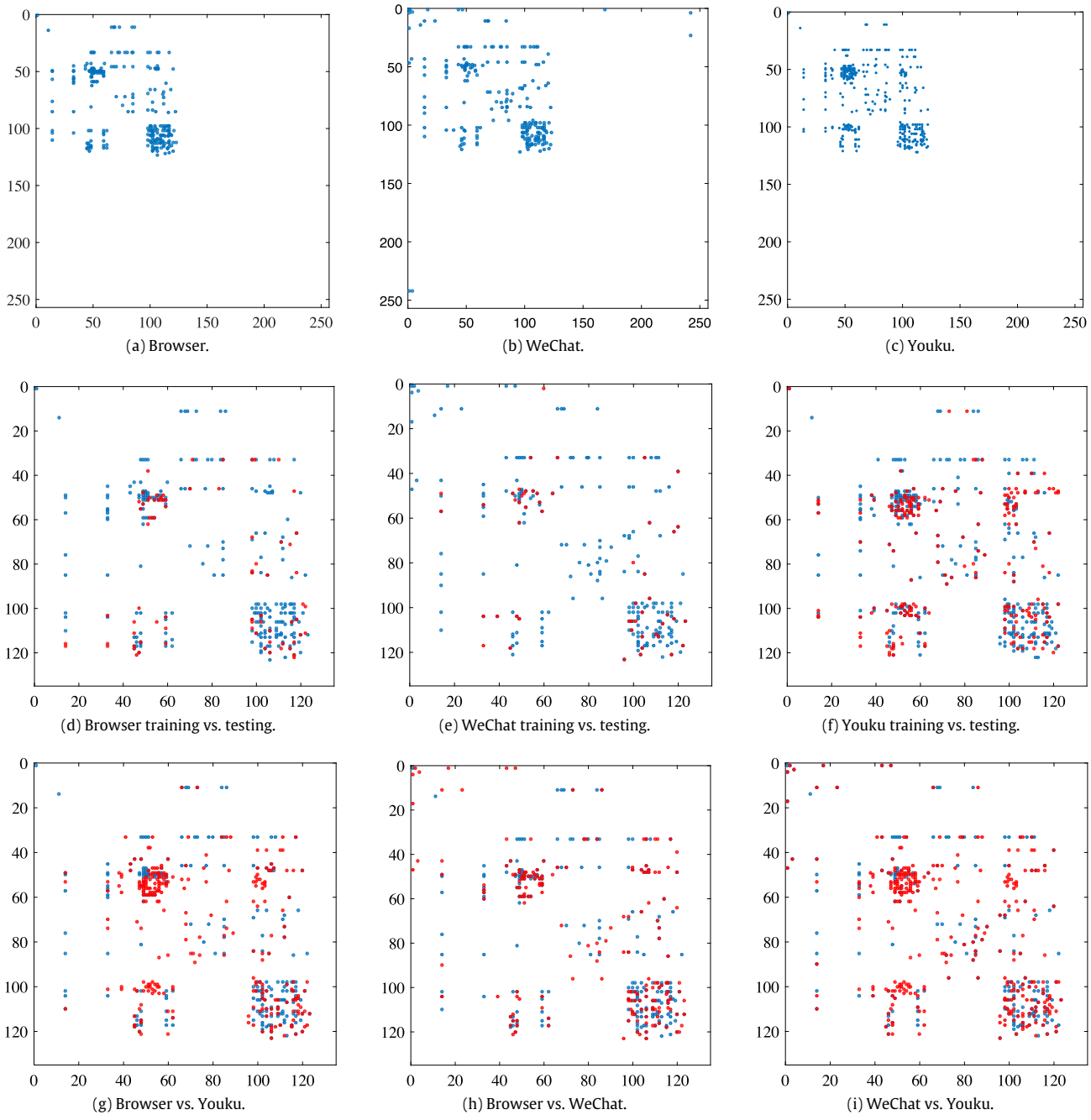
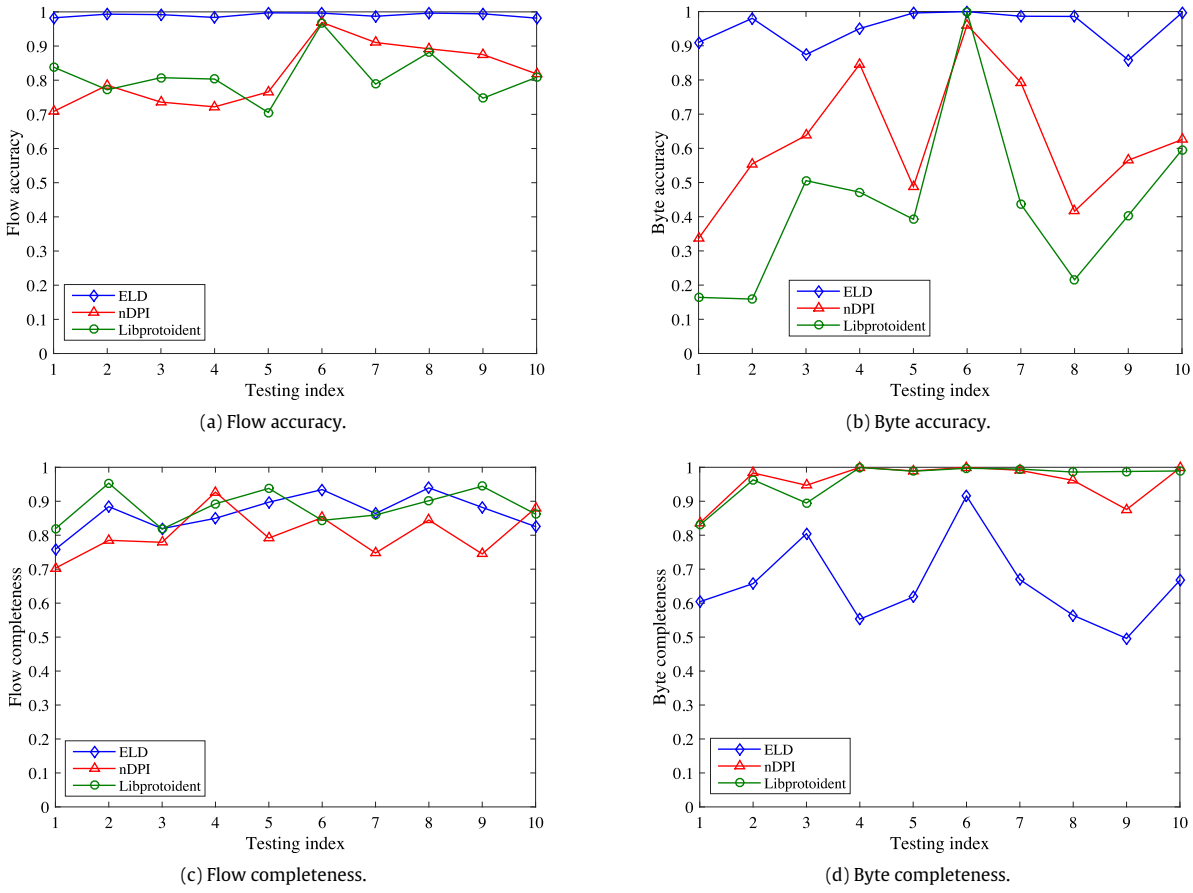



**Fig. 11.** Payload distribution of three popular apps. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

**Table 2**
The relationship between protocols and traffic classes.

| Traffic classes | nDPI | Libprotoident |
|---|---|---|
| Web | HTTP.Amazon, HTTP.Google, SSL.Google, MPEG, SOCKS, SSL.IFLIX, HTTP.IFIX, NETBIOS, Web, EAQ, SSL, HTTP.HTTP, HTTP_Proxy, | Web, SPDY, HTTP_NonStandard, HTTP_443, SSL/TLS |
| Streaming | HTTP.QuickTime, HTTP.Flash | Kuaibo, kugou, PPlive,YY_TCP, KankanTCP, PPStream |
| Social | SSL.QQ, HTTP.QQ, SSL.Facebook, HTTP. Sina(Weibo), SSL.Sina(Weibo), Skype | QQ, WeChat, XMPP |
| Mail | SSL.Yahoo | |
| Service | Download, Thunder | Postgresql, SSDP, NTP, STUN, ICMP, Mystery_8000_UDP, BitTorrent_UDP |

NBAR and L7-filter are not suitable for labeling network traffic due to their bad performance [30]. Hence, in this paper, nDPI [10] and Libprotoident [11] are chosen to label mobile traffic data. nDPI and Libprotoident tag the protocols for network traffic. Whereas, ELD




(a) Flow accuracy. (b) Byte accuracy.

(c) Flow completeness. (d) Byte completeness.

**Fig. 12.** Overall identification results obtained by ELD, nDPI and Libprotoident.

**Table 3**
Average flow and byte accuracies obtained by ELD, nDPI and Libprotoident.

| Methods | *Flow acc.* | *Byte acc.* | *Flow com.* | *Byte com.* |
|---|---|---|---|---|
| ELD | **0.990 ± 0.006** | **0.954 ± 0.054** | 0.865 ± 0.055 | 0.655 ± 0.124 |
| nDPI | 0.818 ± 0.140 | 0.622 ± 0.327 | 0.805 ± 0.118 | 0.958 ± 0.109 |
| Libprotoident | 0.812 ± 0.137 | 0.434 ± 0.390 | **0.883 ± 0.090** | **0.963 ± 0.102** |

mainly identifies the applications of mobile traffic. To compare the mobile traffic identification performance, we group the protocols and applications into high level (traffic classes) shown as Table 2.

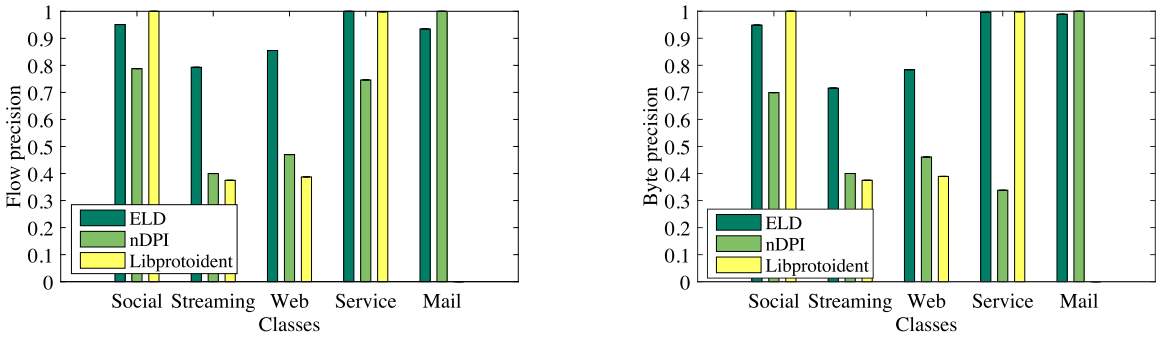
Fig. 12 shows that the performance of ELD is more stable than the other two methods among the 10 independent experiments. ELD always performs the best in terms of flow accuracy, and the flow completeness of ELD is comparable with that of nDPI and Libprotoident. ELD also outperforms others in terms byte accuracy, although its byte completeness is worse than those of nDPI and Libprotoident. ELD aims to achieve high identification accuracy. This target may sacrifice the identification completeness.

The flow accuracy of ELD is often close to 99%, but the byte accuracy is about 85.8% in the worst case. To further analyze the classification results in the worst case, we found that the bye accuracy is decreased at the third stage of ELD, i.e. the Random Forest. This may be resulted by the data drift problem, as the dynamic characteristic of mobile traffic. The traditional machine learning techniques assume that the training and testing sets follow the same distribution. When concept drift happens, the classification performance may be degraded. Therefore, in order to label the mobile traffic in high accuracy using ELD, we suggest the initial labeled data and the data to be labeled should be collected in the same period of time.

The average results are shown in Table 3. nDPI performs better than Libprotoident in terms of byte accuracy, which is in accordance with the results in [30]. ELD achieves more than 95% of flow accuracy and byte accuracy, and it acquires 87.1% flow completeness and 67.2% byte completeness on average.

### 5.2.2. Per class classification performance

The identification results of each class are shown in Fig. 13. ELD performs the best on average. And it significantly outperforms nDPI and Libprotoident on the precisions of Steaming and Web. On the results obtained by ELD, the precisions of Web/Streaming are not as good as others. The possible reasons are analyzed as follows. On the aspect of the performance of Web, some other flows are misclassified as Web by ServerTag. The server providing Social/Streaming service may also support web browsing service. The server is only linked to Social/Streaming on the training set. When more traffic are collected, all services supported by a server could be exposed. ServerTag would pass the traffic generated by the server with more than one service to the next stage, so as to be avoid being wrongly labeled by ServerTag. On the aspect of the performance of Streaming, the payload distribution in Fig. 11 shows that the payload distribution of Youku (Streaming class) is more random than that of other apps, leading to that the Streaming flows cannot be well handled by PDI.




(a) Flow precision of each class.

(b) Byte precision of each class.

**Fig. 13.** Per class identification results.

**Table 4**
Results on identifying special flows.

| Performance metrics | Results |
|---|---|
| Accuracy of identified encrypted flows | $0.970 \pm 0.032$ |
| Accuracy of identified flows without payload | $0.967 \pm 0.042$ |
| Completeness of identified encrypted flows | $0.669 \pm 0.137$ |
| Completeness of identified flows without payload | $0.526 \pm 0.243$ |

**Table 5**
The identification results of ELD in different stages.

| *Stages* | *Flow acc.* | *Flow com.* | *Byte acc.* | *Byte com.* |
|---|---|---|---|---|
| ServerTag | 0.997 | 0.771 | 0.992 | 0.395 |
| PDI | 0.993 | 0.857 | 0.978 | 0.627 |
| Random forest | 0.990 | 0.865 | 0.954 | 0.655 |

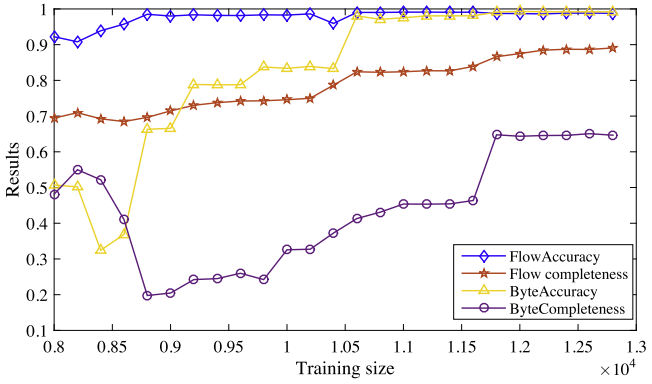



**Fig. 14.** Overall identification results of ELD with different training sizes.

#### 5.2.3. Performance on special flows

ELD is able to handle encrypted traffic and the traffic without payload, because ServerTag and Random Forest do not rely on packet payload. The identification results on the special flows are shown in Table 4. ELD achieves more than 96% accuracy on identifying the special flows on average. ELD obtains 92% accuracy in the worst case on identifying encrypted flows, and 87% in the worst case on identifying flows without payload. As mentioned above, the concept drift may degrade the performance of Random Forest. Searching a set of robust flow features is still an open problem to improve the robustness of machine learning techniques in mobile traffic classification field.

### 5.3. Discussion on ELD

#### 5.3.1. Stage discussion

ELD includes three stages to predict the labels of unknown mobile traffic. The identification results after each stage are shown in Table 5. In the first stage, ServerTag identifies the HTTP traffic according to the association between servers and traffic classes. After this stage, our approach achieves 99.7% flow accuracy and 99.2% byte accuracy, but the flow completeness and byte completeness are only 77.1% and 39.5% respectively. To extend the labeled mobile traffic, PDI and Random Forest further identify unknown traffic. After the two stages, flow accuracy and byte accuracy are maintained, and the flow completeness and byte completeness are increased to 86.5% and 65.5% respectively.

#### 5.3.2. Training set size

We take the last five days traffic as testing sets, and others as training sets. The identification results with different training set sizes are shown in Fig. 14. As increasing the number of training flows. Flow and byte accuracies are increasing progressively. Although flow completeness and byte completeness are fluctuate at beginning, both of them are improved at last.

#### 5.3.3. Discussion on parameters of ELD

There are two parameters in PDI. The first $K$ bytes for n-gram generation and the threshold $\alpha$ for labeling each flow.

**(1) First $K$ bytes in PDI**

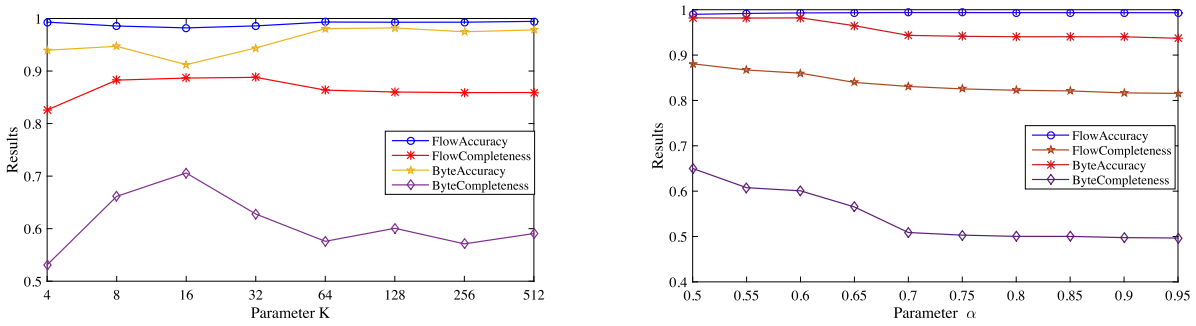
The $K$ is selected in [4, 8, 16, …, 512]. The identification results shown in Fig. 15(a) are summarized in terms of flow accuracy, flow completeness, byte accuracy and byte completeness. As increasing $K$, the identification accuracy tends to be improved and be stable. The completeness tends to be decreased because less traffic are identified while more payload bytes are used to build BPDPs. When $K$ equals to 128, byte accuracy and flow accuracy are the highest.

**(2) The threshold in PDI**

The threshold $\alpha$ in PDI is selected in [0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95]. The identification results are summarized in Fig. 15(b). Flow accuracy is increased while increasing $\alpha$. Byte accuracy and byte completeness are decreased when further increasing $\alpha$. The higher $\alpha$ is, less traffic will be identified by PDI. The ratio of correctly identified bytes decreases a little when increasing $\alpha$. This may be resulted by a few misclassified elephant flows.

**(3) Machine learning techniques**

Machine learning based traffic identification methods have been the hot topic in recent years [28]. We will carry out experiments to evaluate the identification performance of machine learning techniques on our mobile traffic data by 10-fold cross validation. The machine learning techniques in the study include C4.5 decision tree, Naïve Bayes Kernel (NBK), Neural network, Nearest Neighbor, Random Forest, AdaBoostM1and Bagging. All machine learning algorithms are implemented by Weka toolkit [31], and default parameter values set by Weka are used in our experiments. Overall identification results are shown in Table 6. The ensemble classifiers (Random forest, AdaboostM1 and Bagging) perform better than others. And Random Forest performs the best.




(a) Parameter $K$. (b) Parameter $\alpha$.

**Fig. 15.** Overall identification results of ELD with different values of $K$ and $\alpha$.

**Table 6**
Overall identification results using different classifiers.

| Classifiers | *Flow acc.* | *Byte acc.* |
|---|---|---|
| C4.5 | 0.864 ± 0.008 | 0.753 ± 0.080 |
| NBK | 0.490 ± 0.032 | 0.294 ± 0.082 |
| Neural network | 0.690 ± 0.012 | 0.446 ± 0.062 |
| Nearest neighbor | 0.796 ± 0.010 | 0.765 ± 0.049 |
| Random forest | **0.905 ± 0.006** | **0.869 ± 0.070** |
| AdaBoostM1 | **0.905 ± 0.006** | 0.860 ± 0.074 |
| Bagging | 0.895 ± 0.006 | 0.836 ± 0.086 |

The flow and byte $F$-measures of each class are shown in Table 7. $F$-measure is the composite evaluation of recall and precision, i.e. $F\text{-}measure = 2 * Recall * Precision / (Recall + precision)$. The flow (or byte) recall of a class refers to the ratio between correctly classified flows (or bytes) of the class and the total number of flows (or bytes) in the class. The definitions of flow (or byte) precision can be found in Section 4.2. Table 7 shows that the ensemble classifiers still perform better in terms of F-measure for most classes. Random forest performs the best in most cases. Therefore, it is applied in ELD to implement flow statistical feature based mobile traffic identification.

**(4) Parameter $\beta$ in traffic identification using Random Forest**

Random Forest is implemented by Weka toolkit, and the default parameters are used in our experiments. When labeling an unknown flow, ELD applies a threshold $\beta$ to select out the traffic flows to be labeled. The threshold $\beta$ is set in [0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95]. The identification results with different values of $\beta$ are shown in Fig. 16. As increasing $\beta$, flow accuracy and byte accuracy are also increased. But flow completeness and byte completeness are decreased, because less traffic flows are identified as increasing $\beta$. When ELD is deployed on real cases, $\beta$ is chosen to be higher if the system aims to obtain high identification accuracy, such as in the range of [0.9 0.95].

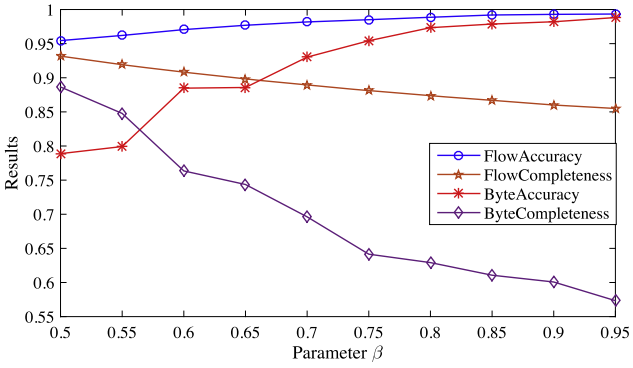



**Fig. 16.** Overall identification results of ELD with different values of $\beta$.

### 5.4. Simulation on labeling newly unknown traffic

A usage case of ELD is shown in Fig. 17. *MgtClient* is installed on some mobile nodes, and *mgtServer* is deployed on a server to collect the traffic data generated by these nodes and link ground truth to them, so as to obtain initial labeled data. On the edge router of the local network, the raw traffic data are mirrored to a server. And then, ELD extends the labeled data by labeling more unlabeled traffic data.

This section simulates the usage of ELD in real network environment. To evaluate the performance of ELD in such case, we collected traffic from newly three nodes during March 2017 to April 2017 in the same network. After data preprocessing, 48 available datasets were obtained from different dates and mobile devices. The basic information of the new traffic data is shown in Fig. 18. The flow and byte distributions are different from that on training

**Table 7**
$F$-measure results using different classifiers.

| Classes | C4.5 | NBK | NeuralNetwork | NearestNeighbor | RandomForest | AdaBoostM1 | Bagging |
|---|---|---|---|---|---|---|---|
| *Flow F-measure* | | | | | | | |
| Social | 0.839 ± 0.012 | 0.217 ± 0.085 | 0.640 ± 0.022 | 0.752 ± 0.014 | **0.889 ± 0.010** | 0.888 ± 0.011 | 0.878 ± 0.011 |
| Streaming | 0.703 ± 0.022 | 0.356 ± 0.060 | 0.515 ± 0.048 | 0.643 ± 0.034 | 0.777 ± 0.020 | **0.787 ± 0.012** | 0.761 ± 0.021 |
| Web | 0.834 ± 0.014 | 0.424 ± 0.098 | 0.645 ± 0.024 | 0.747 ± 0.019 | **0.881 ± 0.012** | 0.878 ± 0.012 | 0.869 ± 0.014 |
| Service | 0.949 ± 0.006 | 0.665 ± 0.050 | 0.821 ± 0.008 | 0.923 ± 0.007 | 0.969 ± 0.005 | **0.970 ± 0.004** | 0.962 ± 0.004 |
| Mail | 0.776 ± 0.034 | 0.217 ± 0.052 | 0 ± 0 | 0.582 ± 0.040 | **0.832 ± 0.035** | 0.830 ± 0.034 | 0.809 ± 0.039 |
| *Byte F-measure* | | | | | | | |
| Social | 0.700 ± 0.135 | 0.167 ± 0.212 | 0.426 ± 0.165 | 0.725 ± 0.116 | **0.850 ± 0.131** | 0.839 ± 0.134 | 0.796 ± 0.151 |
| Streaming | 0.731 ± 0.103 | 0.133 ± 0.238 | 0.053 ± 0.089 | 0.696 ± 0.130 | **0.830 ± 0.085** | 0.815 ± 0.095 | 0.815 ± 0.149 |
| Web | 0.791 ± 0.058 | 0.104 ± 0.099 | 0.545 ± 0.060 | 0.781 ± 0.053 | **0.891 ± 0.029** | 0.870 ± 0.046 | 0.862 ± 0.048 |
| Service | 0.774 ± 0.191 | 0.387 ± 0.086 | 0.476 ± 0.134 | 0.839 ± 0.066 | 0.885 ± 0.130 | **0.891 ± 0.133** | 0.869 ± 0.131 |
| Mail | 0.718 ± 0.164 | 0.144 ± 0.052 | 0 ± 0 | 0.294 ± 0.079 | 0.778 ± 0.178 | **0.795 ± 0.162** | 0.726 ± 0.176 |

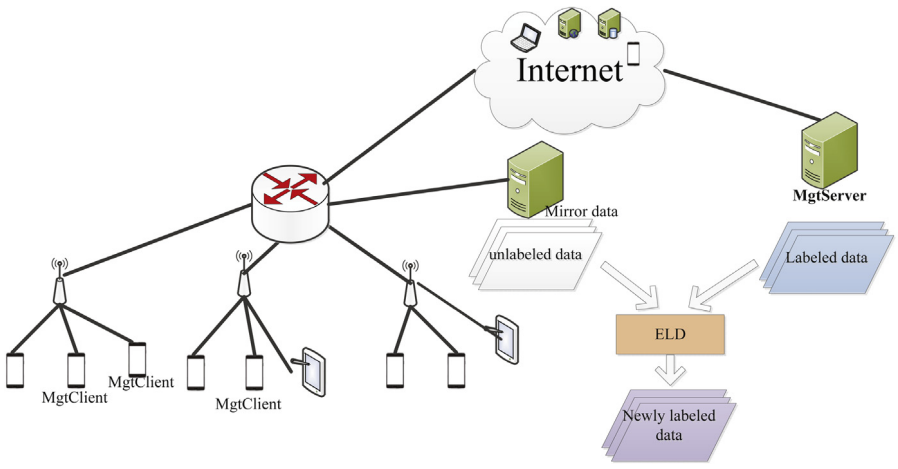



**Fig. 17.** Usage case of ELD.

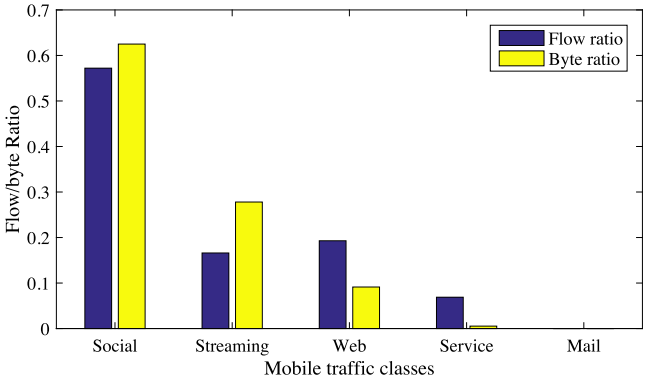



**Fig. 18.** Distribution of newly labeled data.

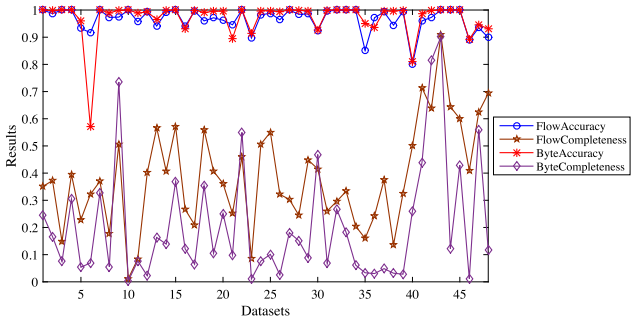



**Fig. 19.** Overall identification results on newly collected traffic data.

set shown in Section 4.2. In overall, Social owns much more flows and byte than others.

The model trained by the experimental datasets shown in Section 4.1 are used to label the newly collected traffic data. The overall identification results are shown in Fig. 19. ELD achieves more than 90% flow and byte accuracies on most testing datasets, which is much better than the performance reported in [32]. The results in [32] show that their model obtains less than 70% accuracy when identifying the mobile traffic generated by unseen mobile devices. On average, ELD achieves 96.6% flow accuracy, 38.3% flow completeness, 97.0% byte accuracy and 20.5% byte completeness.

In the worst case, the byte accuracy is only about 57% on the sixth testing set. On this dataset, ServerTag obtains low byte accuracy. Similar to the analysis in Section 5.2.2, one way to handle this problem is to collect more labeled traffic. Another way is to limit the usage of ServerTag, i.e. only the collection time of testing data is close to that of training data, the unknown traffic could be labeled by ServerTag. For instance, when ServerTag is not adopted to label the newly collected data, the flow accuracy, flow completeness, byte accuracy and byte completeness obtained by ELD are respectively 99.9%, 9.6%, 99.93%, 2.8% on average. Flow accuracy and byte accuracy are respectively 96.3% and 97.3% in the worst case. Although the completeness is decreased significantly, both flow accuracy and byte accuracy are much high. This results are acceptable when pursuing high precision for extending labeled traffic data. In addition, as more and more traffic are generated every day, unknown mobile traffic data are easily obtained.

## 6. Conclusion

This paper handles the problems of collecting labeled data for mobile network traffic classification. Conventional methods suffer from poor performance on labeling encrypted traffic or traffic limitation on simulation network environment. This paper presents a method named ELD to enlarge the scale of labeled traffic automatically at the base of initial labeled data. And it is able to label the encrypted traffic and the traffic without payload. ELD concludes three models, respectively, ServerTag, PDI and Random Forest. These models identify unknown mobile traffic from different levels, respectively, packet header level, payload header level and flow statistic level. The cross validation results show that ELD obtains high identification accuracy and significantly outperforms nDPI and Libprotoident. Moreover, ELD acquires more than 96% flow and byte accuracies on average when identifying newly collected traffic data. The 30 experimental datasets can be found in [29]. These data include popularly used mobile apps in China. The performance on identifying Streaming and Web could be further improved by limiting the usage of ServerTag and extracting robust flow features and payload features in future.

## Acknowledgments

We thank the anonymous reviewers for their constructive comments. This work was supported by National Natural Science Foundation of China under Grant No. 61501128, financial support from China Scholarship Council, supported by Guangdong Provincial Natural fund project, China (Nos. 2017A030313345, 2016A030310300, 2014A03031358), the Specialized Fund for the Basic Research Operating expenses Program of Central College (No. x2rj/D2174870), and Guangdong Province Youth Innovation Talent Project (2014KQNCX139).

## References


[1] Y. Fu, H. Xiong, X. Lu, et al., Service usage classification with encrypted Internet traffic in mobile messaging apps, IEEE Trans. Mob. Comput. 15 (11) (2016) 2851–2864.
[2] D. Naboulsi, M. Fiore, S. Ribot, et al., Large-scale mobile traffic analysis: a survey, IEEE Commun. Surv. Tutor. 18 (1) (2016) 124–161.
[3] X. Yun, Y. Wang, Y. Zhang, et al., A semantics-aware approach to the automated network protocol identification, IEEE/ACM Trans. Netw. 24 (1) (2015) 583–595.
[4] l7-filter[Online], available: http://l7-filter.sourceforge.net.
[5] A. Tongaonkar, A look at the mobile app identification landscape, IEEE Internet Comput. 20 (4) (2016) 9–15.
[6] J. Zhang, X. Chen, Y. Xiang, et al., Robust network traffic classification, IEEE/ACM Trans. Netw. 23 (4) (2015) 1257–1270.
[7] M. Conti, L.V. Mancini, R. Spolaor, et al., Analyzing android encrypted network traffic to identify user actions, IEEE Trans. Inf. Forensics Secur. 11 (1) (2016) 114–125.
[8] Z. Liu, R.Y. Wang, D.Y. Tang, et al., A system for linking ground truth to mobile network taffic. in: Proc. MOBIQUITOUS, 2016, pp. 292–293.
[9] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[10] L. Deri, M. Martinelli, T. Bujlow, A. Cardigliano, nDPI: open-source high-speed deep packet inspection, in: Proceedings of the 10th International Wireless Communications & Mobile Computing Conference, 2014, pp. 617–622.
[11] S. Alcock, R. Nelson, Libprotoident: traffic classification using lightweight packet inspection, Tech. rep., University of Waikato, 2012.
[12] T.T.T. Nguyen, G. Armitage, A survey of techniques for internet traffic classification using machine learning, IEEE Commun. Surv. Tutor. 10 (3) (2008) 56–76.
[13] G. Ranjan, A. Tongaonkar, R. Torres, Approximate matching of persistent lexicon using search-engines for classifying mobile app traffic, in: INFOCOM, 2016, pp. 1–9.
[14] P. Casas, P. Fiadino, A. Bar, IP mining: extracting knowledge from the dynamics of the Internet addressing space, in: Teletraffic Congress, 2013, pp. 1–9.
[15] H.Y. Yao, G. Ranjan, A. Tongaonkar, et al., SAMPLES: self adaptive mining of persistent lexical snippets for classifying mobile application traffic, in: Proc. of the 21st Annual International Conference on Mobile Computing and Networking, 2015, pp. 439–451.
[16] Q. Xu, Y. Liao, S. Miskovic, et al., Automatic generation of mobile app signatures from traffic observations, in: Proc. of IEEE Infocom, 2015, pp. 1481–1489.
[17] X. Han, Y. Zhou, L. Huang, et al., Maximum entropy based IP-traffic classification in mobile communication networks, in: IEEE Wireless Communications and Networking Conference, WCNC, 2012, pp. 2140–2145.
[18] A. Satoh, T. Osada, T. Abe, et al., Traffic classification in mobile IP network, in: IEEE International Conference on Ubiquitous Information Technologies & Applications, 2009, pp. 1–6.
[19] S. Mongkolluksamee, V. Visoottiviseth, K. Fukuda, Enhancing the performance of mobile traffic identification with communication patterns, J. Inf. Process. 24 (2) (2016) 247–254.
[20] F. Gringoli, L. Salgarelli, M. Dusi, et al., GT: picking up the truth from the ground for internet traffic, ACM Sigcomm Comput. Commun. Rev. 39 (5) (2009) 12–18.
[21] Z. Liu, R.Y. Wang, D.Y. Tang, Research on mobile network traffic taxonomy, in: IEEE International Conference on Computer, Information and Telecommunication Systems, 2016, pp. 1–6.
[22] P. Holland, Can Passive Mobile Application Traffic be Identified Using Machine Learning Techniques (Master thesis), Dublin Institute of Technology, 2015.
[23] M. Suzuki, M. Watari, S. Ano, et al., Traffic classification on mobile core network considering regularity of background traffic, in: Proc. of the IEEE International Workshop Technical Committee on Communications Quality and Reliability, 2015, pp. 1–6.
[24] T. Bujlow, P. Barlet-Ros, Independent comparison of popular DPI tools for traffic classification, Comput. Netw. 76 (2015) 75–89.
[25] Z. Liu, R. Wang, M. Tao, et al., A class-oriented feature selection approach for multi-class imbalanced network traffic datasets based on local and global metrics fusion, Neurocomputing 168 (C) (2015) 365–381.
[26] M. Scigocki, S. Zander, Improving machine learning network traffic classification with payload-based features, CAIA Technical Report 131120A, 2013, pp. 1–7.
[27] B. Park, Y. Won, J.Y. Chung, et al., Fine-grained traffic classification based on functional separation, Int. J. Netw. Manag. 23 (5) (2013) 350–381.
[28] J. Zhang, Y. Xiang, Y. Wang, et al., Network traffic classification using correlation information, IEEE Trans. Parallel Distrib. Syst. 24 (1) (2013) 104–117.
[29] Labeled mobile traffic data, available at https://wangruoyu.github.io/mobileg t/.
[30] V. Carela-Español, T. Bujlow, P. Barlet-Ros, Is our ground-truth for traffic classification reliable? Passive Act. Meas. (2014) 98–108.
[31] Weka toolkit, available at http://weka.wikispaces.com/home.
[32] H.F. Alan, J. Kaur, Can android applications be identified using only TCP/IP headers of their launch time traffic? in: ACM Conference on Security & Privacy in Wireless and Mobile Networks, 2016, pp. 61–66.


**Zhen Liu** received the Ph.D. degree from the School of Computer Science and Technology of South China University of Technology, China, in 2013. She received her Bachelor's degree from Department of Computer Science and Technology of South West University, China in 2008. She is now a Lecturer in the School of Medical Information Engineering, Guangdong Pharmaceutical University, Guangzhou, China. She is a member of CCF (China Computer Federation). She serves as a reviewer of Neurocomputing and International Journal of Communication Systems. Her research interests are in the areas of mobile network traffic classification and machine learning.

**Ruoyu Wang** received the Ph.D. degree from the school of Computer Science and Engineering, South China University of Technology, China in 2015. He is now an engineer at the Information and Network Engineering and Research Center, South China University of Technology, China. He is a member of CCF (China Computer Federation). He serves as a reviewer of Applied Soft Computing and ISA Transactions. His research interests are in the areas of machine learning and complex network.

**Deyu Tang** received the Ph.D. degree from the School of Computer Science and Technology of South China University of Technology, China, in 2015. He is now an associate professor in the School of Medical Information Engineering, Guangdong Pharmaceutical University, Guangzhou, China. He serves as a reviewer of Information Science and Applied Soft Computing. His research interests are in the areas of swarm intelligence and machine learning.