Information Sciences 479 (2019) 526–541

Contents lists available at ScienceDirect

Information Sciences

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

Novel dynamic multiple classification system for network traffic

Xi Xiao[a], Rui Li[a], Hai-Tao Zheng[a,*], Runguo Ye[b], Arun KumarSangaiah[c], Shutao Xia[a]

[a] Graduate School at Shenzhen, Tsinghua University, Shenzhen 518055, China
[b] China Electronics Standardization Institute, Beijing, China
[c] School of Computing Science and Engineering, VIT University, Vellore, India

A R T I C L E   I N F O

Article history:
Received 21 August 2017
Revised 17 October 2018
Accepted 23 October 2018
Available online 24 October 2018

Keywords:
Multiple classification
Topic model
Machine learning
Traffic classification
Network security

A B S T R A C T

Traffic classification has been widely used in networking and security issues. Previous works have involved many different techniques for mapping traffic to the application. However, little attention has been paid to traffic classification for dynamic network stream. In this paper, we propose a Dynamic Multiple Traffic Classification System (DMTCS). We first introduce the time-based distribution of the traffic protocol information to the traffic classification problem, as the traffic data is a data stream with time continuity. The packets are treated as documents and protocols are seen as topics. Thus, we can apply topic models to cluster packets. In our system, after initialization, packets arrived at a time point are classified as of some protocols. Then, these packets are assembled to clusters according to the protocol distribution at the last time point. Finally, we use these clusters to classify packets arrived at the next time point. Our method has several advantages: 1) does not require the prior knowledge of target applications; 2) tolerant with both TCP and UDP protocols; 3) support multiple classification; 4) preserve high accuracy for the traffic stream with dynamic and imbalanced traffic distribution. Evaluations on DMTCS are carried on two different datasets, and the experimental results demonstrate that DMTCS has an impressive performance in classification on the real-world network stream and the dynamic simulation stream. Whats more, DMTCS outperforms other state-of-the-art models in our experiment.

1. Introduction

Nowadays, network traffic classification has drawn plenty of attention [19,20,26,30,32,37]. Mapping the network traffic to the network application is very significant in the area of network security and management [30], such as intrusion detection, Quality of Service (QoS), network measurement and prevention systems. Besides, network traffic analysis is very useful on network security [6] and other areas. Traditional methods [4,19,23,29,31,38] in this field fall into two categories: 1) port-based methods, and 2) datagram format-based methods. While in the current network environment, the traditional methods suffer from many problems. Most network applications have neither a specific port number nor a well-documented format in public. Besides, nearly 50% of internet traffic are unknown applications, according to a report of Internet2 NetFlow

* Corresponding author.
E-mail address: zheng.haitao@sz.tsinghua.edu.cn (H.-T. Zheng).

https://doi.org/10.1016/j.ins.2018.10.039
0020-0255/© 2018 Published by Elsevier Inc.

on backbone traffic [33]. Moreover, there are many botnet protocols which do not have protocol specifications from their designers.

Recent researches have resort to machine learning methods for traffic classification [14,38]. The existing methods include payload-based methods and flow-based methods. The former approaches pay attention to patterns of traffic packets of the specific protocol, while the latter techniques focus on the information and feature of network flow. And there are still some methods trying to combine them. Using network flow features in traffic classification needs to transform traffic packets to flow, which causes more dealing processes. Whats more, incomplete flow is also a problem. Thus, we focus on classifying traffic which only uses packets information.

In the previous works, little attention has been paid to dynamic multiple classification for traffic stream. For a real network node, the traffic through it is continuous in time, and repeating patterns of packet arrive are very useful [22]. Thus, dynamic classification can take advantage of prior information and may achieve higher accuracy. Dynamic multiple traffic classification can be used in network nodes for traffic monitoring or measurement. There are two critical challenges in this problem. One is the dynamic nature of the network traffic, as the categories of protocols and the number of packets of one protocol change over time. The other is that the packets of a specific protocol may have different features. In this work, we take a particular interest in dynamic multiple classification for traffic stream.

To overcome the two challenges, we present two hypotheses: 1) The distribution of the network traffic protocols at time t is related to that at time t-1; 2) A protocol can generate several kinds of packets and the payloads in different types of packets are different. According to these hypotheses, we propose a Dynamic Multiple Traffic Classification System (DMTCS) based on the clustering topic model [24]. We transform the network packet payload to documents and then assemble them to some clusters according to the protocol distribution at the last time point. Every cluster is assigned to a class to which the most packets in the cluster belong. Then, when a new packet is assembled to a cluster, it will be classified to the class to which the cluster is assigned. After initialization, our system can do traffic classification and update itself as time goes. In our system, the distribution information of protocols at time t-1 is used to cluster the packets at time t, and different clusters can be assigned to one protocol.

Our method is evaluated by experiments on two traffic stream datasets with packets caught from the real world. The result shows that our system achieves high accuracy for eleven applications in different classification experiments. Furthermore, we compare our method with Securitas [39] and nDPI [9]. Our approach shows advantages on four target applications.

The contribution of this work is threefold:

1. We propose a Dynamic Multiple Traffic Classification System (DMTCS) that can classify network traffic dynamically and automatically.
2. We apply a dynamic cluster model of short documents in traffic classification and use previous network traffic distribution information to improve system ability.
3. We thoroughly analyze the effectiveness of the traffic classification model by testing it on several traffic protocols and in different traffic environments with two datasets and compare our method with two methods proposed in recent years.

The remainder of the paper is organized as follows: Section 2 discusses related work; Section 3 gives the background; Section 4 describes the proposed network classification system; Section 5 details our experimental results, and we conclude the paper in Section 6.

2. Related work

In recent years, research about network traffic classification has involved many different techniques. As we described above, these methods can be classified into two categories: 1) flow-based methods, 2) payload-based methods.

The flow-based methods deal with traffic classification in the flow layer. In flow-based methods, the features and the behavior of network flow are used for classification. Zhang et al. proposed Traffic Classification using Correlation (TCC) in [41], which incorporated flow correlated information into classification. However, in their method, the network packets must be transformed to traffic flow, and the correlated information is obtained from the flow. Zhang et al. also presented an approach to tackle the problem of unknown applications [40]. This approach tried to sufficiently utilize flow correlation information by flow label propagation and compound classification techniques. The works of [34] and [42] used both flow information and payload information in traffic classification. These approaches treated payload signatures and information of flow as properties for the supervised or unsupervised method. Meanwhile, there were some flow-based methods showing interests in traffic behavior. Karagiannis et al. observed and identified patterns of host behavior in the transport layer [19]. Bernaille et al. found out that the size and the direction of the first few packets of the TCP connection can be leveraged to recognize the behavior of an application [2]. Iliofotou et al. [16] tried to represent network traffic with series of related graph instances that vary over time and utilize the representation to analyze the dynamic nature of network traffic. Huang et al. proposed APPlication Round method (APPR) [15] to identify network traffic at the early stage by characterizing the possible negotiation behaviors of every flow. Moreover, Ducange et al. paid attention to design an interpretable machine learning system and proposed Multi-objective Evolutionary Fuzzy Classifiers [10] to figure out the relation of features.

Although the flow-based methods have different focuses, they all require the network flow information, i.e., they have to reconstruct the network flow. That brings an additional cost to network classification and has a strict request for the training data. Besides, the growth of applications in the real world is so rapid that it is hard to find a series of flow features for every application.

The payload-based techniques also have different emphasizes. Some works focus on parsing the process of packet generation. Lim et al. brought forward an analysis tool to extract the network packet format [25]. The work by Caballero et al. in [5] employed program binaries to produce protocol message format. Whats more, other methods tried to use reverse engineer techniques. The approach in [7] focused on reverse engineer for the protocol state machine. Cui et al. proposed Tupni [8], a tool that can reverse engineer for the input format from record sequences, record types, and input constraints. These two kinds of methods require the executable code of the application and may cost much time. Moreover, it is hard for them to work when facing the packets issued by the unknown application protocol.

At the same time, plenty of researches [12–14,18,21,28] concentrated on generating packet signatures from traffic itself. Haffner et al. presented ACAS [14], which applies three machine learning algorithms to get packet signatures. By taking advantage of IP traffic payload content of the known instance in each protocol, ACAS can automatically extract signatures. Ma et al. proposed a method to identify traffic of the same protocol in [28]. This method took into account the network traffic issued by one unknown application and automatically identified the traffic as distinct from known ones. The approach in [18] tried to extract the apparent structure of application sessions by mining a large corpus of network connection logs. It was semi-automated in presenting an analyst with high-quality information. Finamore et al. put forward an approach [12,13] for UDP traffic, which identified the traffic by automatically inferring payload signatures. Support Vector Machines (SVM) was employed in this method to build the decision engine. Moreover, they defined the concept of Stochastic Packet Inspection (SPI). And the work [21] extended the idea of SPI to support the classification of TCP. Automatically generating packet patterns makes these methods can support more applications, but the dynamic network environment still brings many challenges to them. For example, the application may be updated, and the packet signatures may be changed over time. These dynamic changes will make the packet signatures out of action.

There are still some methods shown their interests in introducing cluster techniques to the traffic classification. Bernaille et al. adopted the K-means clustering algorithm to do the traffic classification [1]. The clustering algorithm can construct clusters and use the training data to label them. Erman et al. tried various clustering algorithms in [11], including DBSCAN and AutoClass. High-purity clusters are the most important for this kind of approaches. Generally, high-purity clusters require the cluster number to be large, which consumes more memory and computing resources.

Recently, since the rapid development of deep learning and neural networks, some researchers involved deep learning methods to traffic classification. Lotfollahi et al. proposed “Deep packet” [27] to identify encrypted traffic and distinguishes VPN traffic. Wang et al. tried to transform different kinds of traffic features to images and then using the convolutional neural network to create classifier [35,36]. Comparing with ordinary machine learning methods, the classifier with deep learning methods can gain high accuracy easily, but the huge burden of training and computing make it hard to be used in the real network.

In addition, there are two works which we use for comparison experiments. Deri et al. developed an implementation of Deep Packet Inspection (DPI) named nDPI [9]. nDPI used string detection without degrading network throughput. The packet header and the payload were used in this approach. It can support over 500 different protocols and achieve high accuracy and efficiency. However, the prior knowledge of protocols must be prepared in nDPI. Yun et al. applied Latent Dirichlet Allocation (LDA) to the traffic classification and presented a network trace-based traffic classification model named Securitas [39]. A protocol was deemed as a language in this model, and LDA was used to get the features of the language for classification methods like SVM, C4.5 decision tree, and Bayes Network. Securitas had an impressive performance for traffic classification and achieved high accuracy in the experiments.

Our approach is a payload-based method for network stream, which extracts protocol signatures dynamically. We involve a dynamic topic cluster model [24] into our system and design a novel framework for the dynamic multiple traffic classification system.

3. Background

In this section, we introduce two lines of works related to our method, packet-document generation, and topic models. Both of them have rich literature, and we only discuss the most related models and their applications in traffic classification as follows.

3.1. Packet-document generation

Before applying the topic models to traffic classification, we have to convert network packets to documents. These documents are called as packet-documents in our work. As we know, raw packets consist of continuous payload in text or binary type, while documents are sequences of words. Hence, it is necessary to format payload to a series of words.

Here comes an *n*-gram method proposed by X. Yun et al. [39]. It is useful and straightforward so that we use it to transform raw packets in our study. An *n*-gram is a subsequence of elements in the original packet. Each packet can be broken up into a set of *n*-grams, and each *n*-gram can be seen as one word. By this way, a raw packet can be converted into a document for the topic model. For example, an example of text type payload POST /search can be transformed to the following *n*-grams:

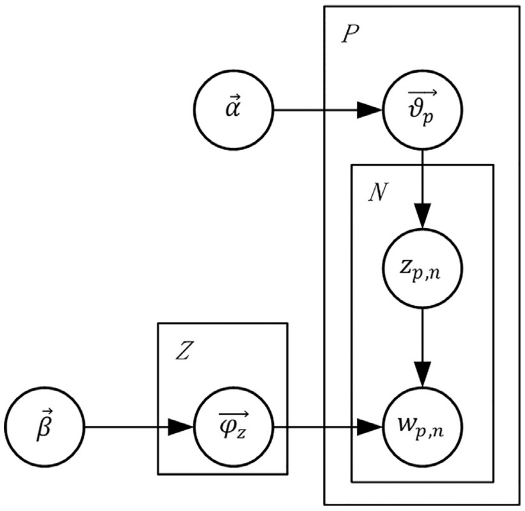

Fig. 1. Structure of LDA.

2-g: PO, OS, ST, T(b), (b)/, /s, se, ea, ar, rc, ch;
3-g: POS, OST, ST(b), (b)/s, /se, sea, ear, rch;
4-g: POST, OST(b), ST(b)/, T(b)/s, (b)/se, /sea, sear, earc, arch; where (b) means the blank character.

This method can be used to transform binary type payload too. An example of binary DNS payload 0xb0/0x61/0x01/0x00/0x00/0x01 can be broken up to 3-g like the following:

0xb0x610x01, 0x610x010x00, 0x010x000x00, 0x000x000x01

Like the previous work shows, 3-g can achieve reasonable results with acceptable cost [39]. To this end, we set the *n*-gram value to be 3 in our study.

3.2. Topic model

A topic model is a statistical model to discover the abstract topics that occur in a collection of documents. If a document is relevant to a specific topic, some words may appear in this document more or less frequently. Latent Dirichlet Allocation(LDA) [3,17] is one of the simplest topic models. In LDA, each document is a mixture of many topics which contain a lot of words, and each word is assigned to topics. The structure of LDA is shown in Fig. 1

In Fig. 1, $\vec{\alpha}$ is the parameter of the Dirichlet prior on the per-document topic distribution, and $\vec{\beta}$ is the parameter of the Dirichlet prior on the per-topic word distribution, $\vec{\varphi_z}$is the word distribution for topic *z*. $\vec{\vartheta_p}$ is the topic distribution for document *p*, $z_{p,\,n}$ is the topic for the *n*th word in document p, and $w_{p,n}$ is the specific word. Besides, *P*, *Z*, *N* are the total number of documents, topics, and *n*-grams.

The generative process [3] of a document is:

- From Dirichlet distribution $\vec{\alpha}$ draw topic distribution $\vec{\vartheta_p}$ of document *p*.
- From Multinomial distribution $\vec{\vartheta_p}$ draw topic $z_{p,\,n}$ for the *n*th word in document *p*.
- From Dirichlet distribution $\vec{\beta}$ draw word distribution $\varphi_{\vec{z_{p,n}}}$ of topic $z_{p,\,n}$.
- From Multinomial distribution $\varphi_{\vec{z_{p,n]}}}$ draw word $w_{p,n}$.

In previous works, topic models like LDA can extract the signature of protocol payload [39]. Even if the structure of the protocol is unknown, the topic model can still automatically generate the signature. This signature can be seen as the feature of the target protocol, and thus traffic classification can be transformed into a typical classification problem. This method works well for traditional protocols, but cannot adapt to some complex new protocols in practice. In this paper, we propose a new application of the topic model for traffic classification. Our work is based on the topic model Dynamic Cluster Topic (DCT) [24] which is applied to clustering the packet-documents. We design a dynamic classification framework that can improve accuracy by using history information. We will describe our approach in following sections in detail.

4. Proposed method

In this section, we provide the structure of our system in detail. At First, we give the workflow of the Dynamic Multiple Traffic Classification System, DMTCS. Then, we focus on the classifying model, and the essential components are explained in detail. Finally, we analyze the computational complexity of the classification system. For readers' convenience, we list in Table 1 the symbols we used in this paper.

Table 1
Symbol list.

Symbol	Definition
\boldmath α	Dirichlet hyperparameter on the per-document topic distribution
\boldmath β	Dirichlet hyperparameter on the per-topic word distribution
\boldmath V	Number of different n-grams used in dynamic cluster
\boldmath M_t	The n-gram collection of packet-documents at time point t
\boldmath P	Total number of packet-documents at time t
\boldmath N_p	Number of n-grams in packet-document p
\boldmath $N_{p,w}$	Number of word w in packet-document p
\boldmath $w_{p,i}$	The ith n-gram word in document p
\boldmath Θ_t	Topic distribution at time t
\boldmath $\vartheta_{t,\,z}$	The probability of topic z at time t
\boldmath Z	Total number of topics
\boldmath L	Iteration count of Gibbs sampling
\boldmath Φ_t	Word distribution over topics at time t
\boldmath $\varphi_{t,\,z}$	Word distribution of topic z at time t
\boldmath z_p	The Most relevant topic for a packet-document p
\boldmath $x_{t,\,z}$	Total number of documents assigned to topic z
\boldmath $y_{t,z,w,-p}$	Number of word w assigned to topic z in all documents expect p
\boldmath $y_{t,z,-p}$	Number of documents attached to z expect p
\boldmath K	Number of clusters
\boldmath $\Psi(.)$	The digamma function
\boldmath $Pr(.)$	Probability

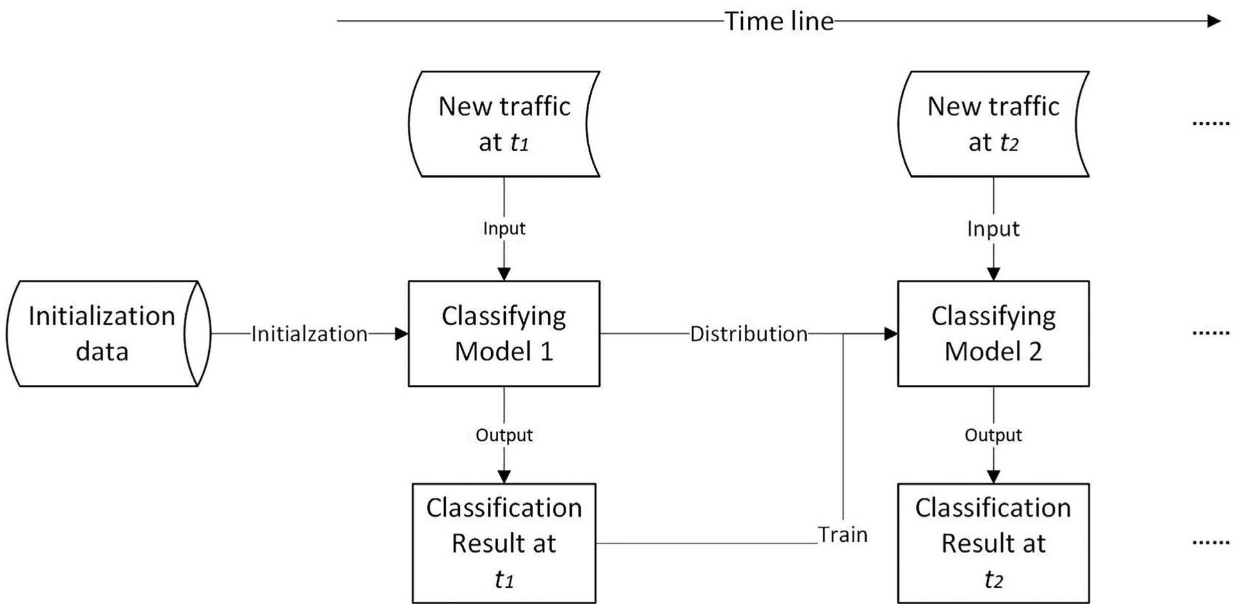

Fig. 2. High-level workflow of DMTCS.

4.1. *Workflow of DMTCS*

High-level workflow of DMTCS is shown in Fig. 2 DMTCS is an online classification system, which can update itself dynamically. In the beginning, DMTCS builds the first classifying model using the initialization data, and this model classifies the new traffic at time t_1. Subsequently, DMTCS constructs the second model using the classification result of time t_1 and the traffic protocol distribution from the clustering processing in the first model. The third model is then created by repeating the operations above. In this way, our system can rebuild itself at every time point t and get the classification results.

4.2. *Architecture of the classifying model*

Fig. 3 shows the architecture of the classifying model in DMTCS. This model has two major phases: 1) training phase; 2) classification phase. Both of them involve some modules. Here comes a brief introduction to them.

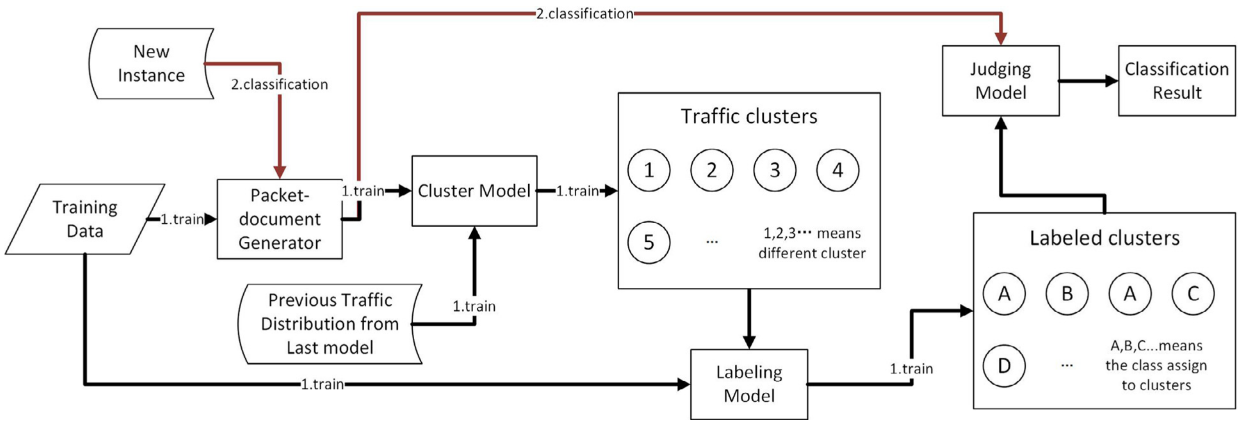

Fig. 3. The architecture of the classifying model.

4.2.1. Training phase

The training phase is the first phase of the classifying model. This phase includes three components: the packet-document generator, the cluster model, and the labeling model. In this phase, the model takes the training data and the traffic protocol distribution from the last model as input and builds the judging model based on labeled clusters. Especially for the first model, the training data is the only input data. This phase can be described as follows. Firstly, the packets from the training data are transformed to documents by the packet-document generator. Then, these documents are assembled by the cluster model according to the traffic protocol distribution from the last model. Finally, by using the labels of packets from the training data, the labeling model assigns a label to every cluster, which represents the class of the cluster. Based on those labeled clusters, the judging model is constructed.

4.2.2. Classification phase

There are two components in the classification phase, i.e., the packet-document generator and the judging model. This phase takes the new packet instance as input and outputs the class to which the packet pertains. In this phase, the new instance is first converted to the packet-document by the packet-document generator and then classified to one labeled cluster by the judging model. As indicated above, every labeled cluster belongs to one class. Accordingly, this instance is classified to the class of the labeled cluster.

4.2.3. Cyclic retraining

As we described above, our system can retrain itself at every time point. And in this paper, we chose to retrain the classifier according to fixed time intervals of τ. For example, when applying our system on the online traffic stream, we will start retraining mechanism after every τ seconds.

In Sections 4.3, 4.4 and 4.6, we provide the details of every component in our model. Besides, we list the theoretical derivation of sampling of cluster model in Section 4.5.

4.3. Packet-document generator

Before using the n-gram method to transform network packets, we remove all the packet headers, which include the Ethernet header, IP header, and transport layer header. These packet headers may contain some unreliable information about the application protocol (such as the target IP), and this information interferes with our model. Thus, we have developed a tool to remove all this part and use the payload. Since our data are saved as pcap files, we analyze the structure of pacp file, and then deal with single packets. The Ethernet header is removed at first. Secondly, the IP header is deleted according to its type (IPv4 or IPv6), the IP options of IPv4 is also removed according to the header length parameter in the header. Finally, we abandon the transport layer header (TCP or UDP). Similarly, we remove the TCP opinion according to the header length in the TCP header.

We also note that, after using n the n-gram method, there are a vast number of unique words in the packet-documents, which cause a considerable burden to the topic model. On the other hand, only a small part of them are related to the format. Thus, it is necessary to reduce the number of different words. To this end, we only choose a part of n-grams with high appearance probability in the packet-documents. In the following discussion, let V denote the n-grams vocabulary size, i.e., the number of different words.

4.4. Cluster model

In this paper, we employ a cluster model for streaming short documents, called Dynamic Cluster Topic (DCT) [24], to establish our cluster model. By using the DCT model, we can cluster our packet-documents and involve previous traffic distribution information to the cluster model. Before introducing the cluster model, we illustrate the mechanism of DCT as follows.

In the DCT model, if no documents arrive, the topic distribution at time t remains the same as the one at time $t-1$. Otherwise, this model is updated when a new set of documents comes at time t. The n-gram collection of packet-documents can be indicated as $M_t = \{\{w_{p,n}\}_{n=1}^{N_p}\}_{p=1}^{P}$, where P is the total number of packet-documents at time t. N_p is the total number of n-grams in packet-document p and $w_{p,n}$ is the nth n-gram in p. All the n-grams in a packet-document correspond to a specific topic z_p, and z_p follows the Multinomial distribution.

Let $\Theta_t = \{\vartheta_{t,z}\}_{z=1}^{Z}$ be the topic distribution at time t, with $\vartheta_{t,z} = Pr(z|t)$ and $\sum_{z=1}^{Z} \vartheta_{t,z} = 1$. And z is the total number of topics. According to DCT, we have,

$$Pr(\Theta_t|\Theta_{t-1}, \alpha) \propto \prod_{z=1}^{Z} \vartheta_{t,z}^{(\alpha_{t,z}\vartheta_{t-1,z})^{-1}} \tag{1}$$

where $\alpha_t = \{\alpha_{t,z}\}_{z=1}^{Z}$, and $\alpha_{t,\,z}$ is topic persistency which represents how salience topic z is at time t compared to that at time $t-1$, and $\vartheta_{t-1,z}$ is the distribution of topic z at time t. In this way, the mean of the current cluster distribution Θ_t depends on the previous distribution Θ_{t-1}.

Let $\Phi_t = \{\varphi_{t,z}\}_{z=1}^{Z}$ be the word distribution over topics at time t and $\varphi_{t,z} = \{\varphi_{t,z,w}\}_{w=1}^{V}$ be the word distribution of topic z at time t, with $\varphi_{t,z,w} = Pr(w|t,z)$ and $\sum_{w=1}^{V} \varphi_{t,z,w} = 1$. V is the n-grams vocabulary size described in Section 4.3. Similarly, we have,

$$Pr(\varphi_{t,z}|\varphi_{t-1,z}, \beta_{t,z}) \propto \prod_{w=1}^{V} \varphi_{t,z,w}^{(\beta_{t,z,w}\varphi_{t-1,z,w})^{-1}} \tag{2}$$

where $\beta_{t,z} = \{\beta_{t,z,w}\}_{w=1}^{V}$, and $\beta_{t,z,w}$ is the persistency of n-gram w in topic z at time t. When time $t=0$, the two distributions above can be initialized as $\vartheta_{0,z} = 1/Z$ and $\varphi_{0,z,w} = 1/V$.

According to the mechanism, we have the generative process for every packet-document at time t as follows:

1. From a Dirichlet prior distribution $\alpha_t\Theta_{t-1}$ draw topic Multinomial distribution Θ_t;
2. From a Dirichlet prior distribution $\beta_{t,z}\varphi_{t-1,z}$ draw word Multinomial distribution $\varphi_{t,\,z}$ for each topic z;
3. For a packet-document $p \in M_t$, draw a topic z_p from Θ_t and for each n-gram in p:
   Draw an n-gram word w_p from Multinomial φ_{t,z_p};

Note that it is necessary to find out which topic is the most relevant topic z_p for a packet-document p at time t. The posterior target distribution is $Pr(z_p|t, p)$, which can be represented as:

$$Pr(z_p|t,p) = \frac{Pr(z_p|z_{t,-p}, p_t, \Phi_{t-1}, \Theta_{t-1}, \alpha_t, \beta_t)}{\sum_{z'_p=1}^{Z} Pr(z'_p|z_{t,-p}, p_t, \Phi_{t-1}, \Theta_{t-1}, \alpha_t, \beta_t)} \tag{3}$$

where $-p$ means all packet-documents expect p. We need to figure out the posterior distribution $Pr(z_p|z_{t,-p}, p_t, \Phi_{t-1}, \Theta_{t-1}, \alpha_t, \beta_t)$, and it can be calculated by the Gibbs sampling, which is the main topic in the next section.

In this cluster model, we treat a packet-document as a mixture of n-grams. According to the hypotheses in Section 1, all the n-grams from a packet-document belong to the same topic. Therefore, after comparing the posterior distribution $Pr(z_p|t, p)$, which topic z_p is the most relevant topic for a packet-document by setting $z_p = argmax_{z_p}(Pr(z_p|t,p))$. Every packet-document can be assigned to one topic, and the set of packet-documents which are assigned to the same topic can be seen as a cluster. By this way, the cluster model assembles these packet-documents to some clusters.

4.5. Gibbs sampling for the cluster model

According to the chain rule, we have such sampling formula for the posterior distribution $Pr(z_p|z_{t,-p}, p_t, \Phi_{t-1}, \Theta_{t-1}, \alpha_t, \beta_t)$:

$$\begin{aligned} &Pr(z_p|z_{t,-p}, p_t, \Phi_{t-1}, \Theta_{t-1}, \alpha_t, \beta_t) \\ &\propto \frac{(x_{t,z} + \alpha_{t,z}\vartheta_{t-1,z})^{-1}}{\sum_{z=1}^{Z} (x_{t,z} + \alpha_{t,z}\vartheta_{t-1,z})^{-1}} \\ &\times \frac{\prod_{w\in V}\prod_{j=1}^{N_{w,p}} (y_{t,z,w,-p} + \beta_{t,z,w}\varphi_{t-1,z,w} + j - 1)}{\prod_{i=1}^{N_p} (y_{t,z,-p} + i - 1 + \sum_{w=1}^{V} \beta_{t,z,w}\varphi_{t-1,z,w})} \end{aligned} \tag{4}$$

where $x_{t,z}$ is the total number of documents in p_t assigned to topic z, $N_{p,w}$ is the number of word w in document p. $y_{t,z,w,-p}$ is the total number of word w assigned to topic z in all documents expect p, and $y_{t,z,-p}$ is the total number of documents attached to z expect p.

Different from the other topic models, α and β are updated during the sampling process. The update rule for α_t and β_t at time t are shown in (5), (6).

$$\alpha_{t,z} \leftarrow \frac{\alpha_{t,z}(\Psi(x_{t,z}+\alpha_{t,z}\vartheta_{t-1,z}) - \Psi(\alpha_{t,z}\vartheta_{t-1,z}))}{\Psi(\sum_{z=1}^{Z}(x_{t,z}+\alpha_{t,z}\vartheta_{t-1,z})) - \sum_{z=1}^{Z}\Psi(\alpha_{t,z}\vartheta_{t-1,z})} \tag{5}$$

where function $\Psi(.)$ is the digamma function;

$$\beta_{t,z,w} \leftarrow \frac{\sum_{z=1}^{Z}\beta_{t,z,v}\varphi_{t-1,z,w}A_{t,z,w}}{\sum_{z=1}^{Z}\varphi_{t,z,w}B_{t,z,w}} \tag{6}$$

where $A_{t,z,w} = \Psi(y_{t,z,w,-p} + \beta_{t,z,w}\varphi_{t-,z,w}) - \Psi(\beta_{t,z,w}\varphi_{t-,z,w})$, $B_{t,z,w} = \Psi(\sum_{w=1}^{V}(y_{t,z,w} + \beta_{t,z,w}\varphi_{t-1,z,w})) - \Psi(\sum_{w=1}^{V}\beta_{t,z,w}\varphi_{t-1,z,w})$ and $y_{t,z,w}$ is the number of word w assigned to topic z in p_t.

When the number of iterations is large enough, the above distribution will converge to the actual distribution. In this way, we can obtain the posterior distribution (4) in Section 4.4.

4.6. Labeling model and judging model

In the labeling model, the clusters which are established by the cluster model and the label information are taken as input. For each cluster, we get the top 20 most common words in the topic and the number of packet-documents belonging to each class. Then for each cluster, the class which has the most number of packet-documents is attached to it. After this, every cluster has a class assigned to it, and these labeled clusters are used in the judging model.

In the judging model, assume that there are K clusters assigned to C different classes. For a new packet-document p_{new} at time t, the posterior distribution $Pr(z_{p_{new}} = k|t, p_{new})$ can be calculated as follows.

$$Pr(z_{p_{new}} = k|t, p_{new}) = \frac{x_{p_{new},k} + \alpha_t}{\sum_{k=1}^{K} x_{p_{new},k} + K\alpha_t} \tag{7}$$

where $x_{p_{new},k}$ is the number of words of p_{new} assigned to cluster k. We can find out which cluster the new packet-document belongs to by $k_{p_{new}} = argmax_{x_k}(Pr(z_{p_{new}} = k|t, p_{new}))$. The class of p_{new} can be set as that of cluster k.

4.7. Computational complexity

Since DMTCS is designed for online traffic classification, we provide the computational theoretic complexity of different phases in this section.

In the training phase, there are three components: the packet-document generator, the cluster model, and the labeling model. The complexity of packet-document generator can be expressed as $\mathcal{O}(P \times h)$, where P is the total number of packet-documents for training and h is the average number of bytes in packets. The clustering complexity is $\mathcal{O}(P \times K \times L)$, where K is the number of clusters, and L is the iteration count of Gibbs sampling. After clustering, labeling model is straightforward and with the complexity of $\mathcal{O}(P)$. In all, the total complexity of the training phase is $\mathcal{O}(P \times K \times L)$, because $K \times L$ is far bigger than h.

In the classification phase, there are two components: packet-document generator and judging model. As above, the complexity of the first components is $\mathcal{O}(P' \times h)$, where P' is the total number of packets need to be classified. In the judging model, the computational complexity can be calculated as $\mathcal{O}(P' \times K \times h)$. Thus, the complexity of classification phase is $\mathcal{O}(P' \times K \times h)$.

5. Evaluation

In this section, we describe our evaluations on DMTCS. At first, we introduce our dataset and metrics. Then, we propose some research questions for our system and show our experiment results according to these questions.

5.1. Dataset

We evaluate our system using two traffic traces, which were captured from the real world in 2014 and 2018. The traces information is listed in Table 2. Trace I is obtained from the campus network. We establish an application processes tracing tool on edge network hosts. In this way, we can achieve packets from any application accurately. Full packets payload is preserved in trace I. For trace II, we collected a 15 hours continuously traffic stream on a local router. Which can be used in our online classifying experiments. For building ground truth of trace II data, we use a deep packet inspection tool which can match packets in the application layer. The deep packet inspection method is based on a packet features library that supports 516 applications, but we still cannot mark packets from new applications precisely in trace II.

Table 2
Details of traces.

Trace	Date	Network / Link	Packets	Volume	Marking method
Trace I	2014-10	campus/ edge	10M	6.75GB	Process tracing
Trace II	2018-6	campus/ edge	40M	32GB	Packet inspection

We use six applications from trace I in experiments, including BitTorrent, DNS, PPLive, Qihoo 360, Sina weibo and SMTP. From trace II, we choose seven applications including DNS, BitTorrent, HTTP, SSL, SSDP, Microsoft, and TeamViewer. In following experiments, all packets from an application are regarded as traffic from one protocol in the application layer, i.e., one application represents a protocol. These target applications consist of both connection-oriented protocols (TCP) and connection-less protocols (UDP). Meanwhile, they contain the payload of both text type and binary type. Besides, both encrypted and unencrypted applications are involved in our experiments.

As the data from trace II is complete traffic stream during a period, we leverage this data to test the performance of our system under real-world network. Since packets from our trace contain more payload information and are marked more accurately, we use these data to test different parameters, simulate the dynamic environment, and provide our comparisons.

In particular, we choose six applications in our trace, composed of three traditional applications, i.e., BitTorrent, DNS and SMTP, and three new applications, i.e., PPLive, Qihoo 360 and Sina weibo. The three traditional applications are well-known protocols, which have been applied in file sharing, domain name services, and e-mail communication. The new applications are responsible for audio and video, safety and social network, which can be used to validate whether our system adapts to new kinds of protocols or not. All the packets are sorted in time order, and in some of our experiments, we mix different packets to simulate the environment of protocol distribution changing as time goes.

5.2. Metrics

Before introducing metrics for our system, we define four metrics for analysis. For each application, let True Positives (TP) be the number of the instances classified to the specific application protocol which are indeed generated by the application, True Negatives (TN) be the number of packets classified to the other application protocols and not made by the particular application. And let False Positives (FP) be the number of packets classified to the specific application protocol but not issued by the specific application, and False Negatives (FN) be the number of packets classified to the other application protocols but generated by the particular application.

The three mainly metrics we used in this work are *Precision*, *Recall*, and $F1 - measure$.

The definition of *Precision* is:

$$Precision = TP/(TP + FP) \tag{8}$$

And *Recall* is defined as:

$$Recall = TP/(TP + FN) \tag{9}$$

$F1 - measure$ can be described as:

$$F1 - measure = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{10}$$

Since our system is a multiple classification system, metrics are defined for every protocol uniquely.

5.3. Research questions

We raise six research questions to find out parameter natures and performance of our system. Question 1–4 are designed to investigate the effort of different parameters, and question 5 is a real-world traffic stream test. Simulation performance of dynamic traffic stream is in question 6, and question 7 introduces our comparison experiments. The research questions we propose for DMTCS are:

RQ1: What is the impact of the n-gram vocabulary size V when applying DMTCS on traffic classification?

RQ2: How does the performance change across different topic number Z on DMTCS?

RQ3: Does the iteration count of Gibbs sampling cause influence on the performance of DMTCS?

RQ4: Is the performance of DMTCS sensitive to the time intervals τ

RQ5: How does DMTCS perform on real-world network stream?

RQ6: Does DMTCS work well when applied to network stream with dynamic and imbalance protocol distribution?

RQ7: How good is the performance of DMTCS compared to the same type traffic classification algorithm proposed in recent years?

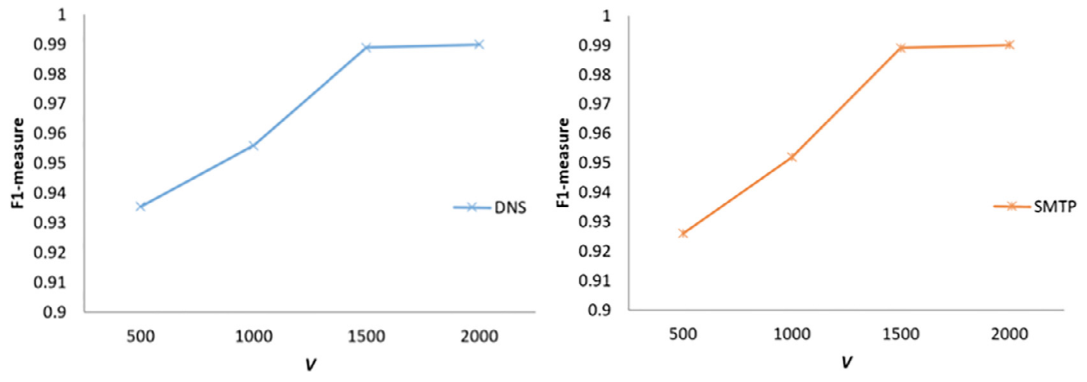

Fig. 4. $F1-measure$ of different vocabulary size for DNS (a) and SMTP (b).

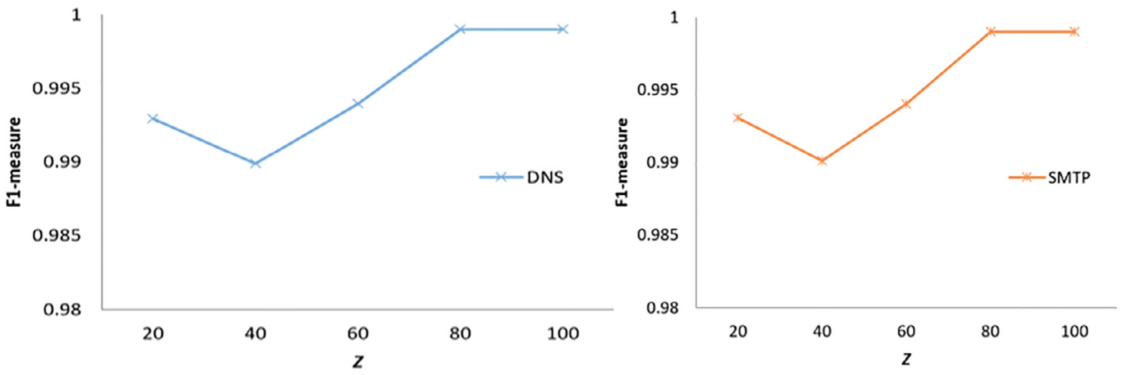

Fig. 5. $F1-measure$ of different topic numbers for DNS (a) and SMTP (b).

5.4. Impact of the n-gram vocabulary size

RQ1: we vary the n-gram vocabulary size V and adopt the average performance of DMTCS to analyze the impact of the vocabulary size on traffic classification.

We select two applications in this experiment, SMTP of text type and DNS of binary type. As discussed in Section 4.3, a subset of n-grams from the traffic with the high probability of appearance should be chosen. We select some subsets with $V = 500, 1000, 1500, 2000$ and then test DMTCSs performance. In this experiment, the topic number Z is 40, and the iteration count is 100.

The performance of DMTCS on the two applications measured by $F1-measure$ is illustrated in Fig. 4. When the vocabulary size V increases from 500 to 1500, the F1-measure of DMTCS becomes better. While after 1500, the F1-measure is nearly unchanged, i.e., it becomes steady. This indicated that a bigger V can improve the performance of DMTCS but when V is large enough, there is almost no improvement. Note that bigger V brings more cost, so the optional V must be as small as possible. In conclusion, V can be the lowest value when the performance of DMTCS becomes steady.

5.5. Effect of the topic number

RQ2: We investigate the performance of DMTCS with different numbers of topics. We vary the topic number Z from 20 to 100 with an interval of 20 and examine the performance of network traffic classification.

In this experiment, we use target applications as same as that in Section 5.4, i.e., DNS and SMTP. The vocabulary size is set as 2000, and the iteration count is set as 100.

Fig. 5 depicts the F1-measure of different topic numbers. It is clear that DMTCS achieves F1-measure higher than 99% for both applications on all the different topic numbers, even the smallest amount 20, which demonstrates our system that can deal with 2-class classification with a small number of topics.

The performance of DMTCS is approximately positive correlated with the topic number Z, and it also becomes steady when Z is big enough. On the other hand, same as the vocabulary size, the big topic number brings enormous costs to our system. Therefore, the tradeoff should be taken into consideration in the selection of Z in our method.

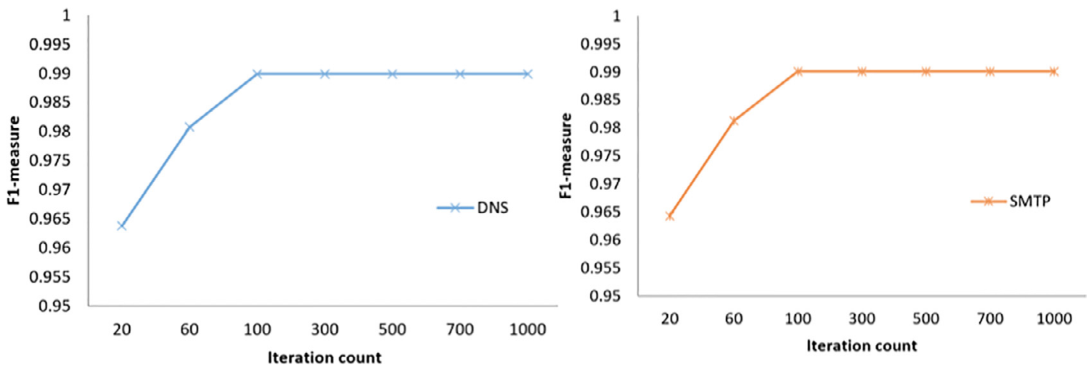

Fig. 6. $F1-measure$ of different iteration count for DNS (a) and SMTP (b).

5.6. Influence of the iteration count

RQ3: We take a deep view of the influence of the iteration count in Gibbs sampling on our system. The iteration count is selected from {20, 60, 100, 300, 500, 700, 1000}.

In this experiment, DNS and SMTP are the target applications. The vocabulary size is set as 2000, and the number of topics is 40.

We adopt an iterative algorithm, the Gibbs sampling, to obtain a sequence of observations from the probability distribution of 3 in Section 4.4. It is crucial to find out an appropriate iteration count for the sampling in our system. Theoretically, when the iteration count is big enough, this method can receive a reasonable result. Fig. 6 shows the performance of DMTCS with different iteration counts. When the iteration count increases from 20 to 100, the F1-measure of DMTCS becomes higher. And it becomes steady after iteration count 100. It can be seen from the figure that our system can reach convergence quickly with a small iteration count. And smaller iteration count can make online classification faster. In this experiment and online classification, the iteration count of 100 is enough.

5.7. Sensitivity analysis of time intervals

RQ:4 For dynamic classification in a realistic setting, choosing different time points to retrain DMTCS may lead to different classification results. To this end, we have some experiments with different time intervals τ on the real-world stream trace II dataset. We have built DMTCS on the same traffic data with time intervals of 0.5, 1, 2, 3, 4 h.

Besides, the vocabulary size is set as 5000, and the iterative count is 100, and the topic number is tuned to 100. In this experiment, we select four target protocols of trace II, which are BitTorrent, HTTP, DNS and SSL.

Fig. 7 shows average $F1-measure$ performances, i.e., average of all classification time points, of DMTCS models with different time intervals τ. Note that data used in this experiment is the real-world stream with fixed time length, Selecting different τ means the number of classification time points is also different. From Fig. 7, we can find out that there is an unnatural point when $\tau = 0.5$ h. When the time interval is 0.5 h, the classifier cannot classify BitTorrent packets and its $F1-measure$ is 0. After checking the experiment data, we figure out that when $\tau = 0.5$, there are only tens of BitTorrent packets are collected, which are not enough for DMTCS to extract its features. When it comes to the other τ and protocols, the performances of DMTCS are relatively stable.

To sum up, the selection of τ must guarantee that every target protocol has enough packets for training. What's more, the performance of DMTCS is not very sensitive to τ, it can be set according to the realistic requirement.

5.8. Real-world stream analysis

RQ:5 We use the trace II dataset to test the performance of DMTCS on real-world network stream. As we have described above, seven target protocols are chosen including DNS, BitTorrent, HTTP, SSL, SSDP, Microsoft, and TeamViewer. The whole traffic data is divided into fifteen parts, and each part holds packets of one hour, which means the retraining parameter $\tau = 3600s$. For every time point, there is one-part traffic data need to be classified. Besides, the data of the first part is used to initialize our system.

Fig. 8 plots the $F1-measure$ for every protocol at different time points in the real-world stream. In this experiment, we set the number of different n-grams V to be 500, the number of topics Z to be 100 and the sampling times L to be 100. It is seen from the figure that for most of the applications, DMTCS achieves high accuracy for many protocols and can keep it as time goes. The traffic data of trace II is very unbalanced and some protocols may have little packets at some time points, such as BitTorrent at time 1–5 and 11–12, which lead to poor performance of DMTCS. Besides, DMTCS has received valuable performance of SSL even though the payload of that is encrypted.

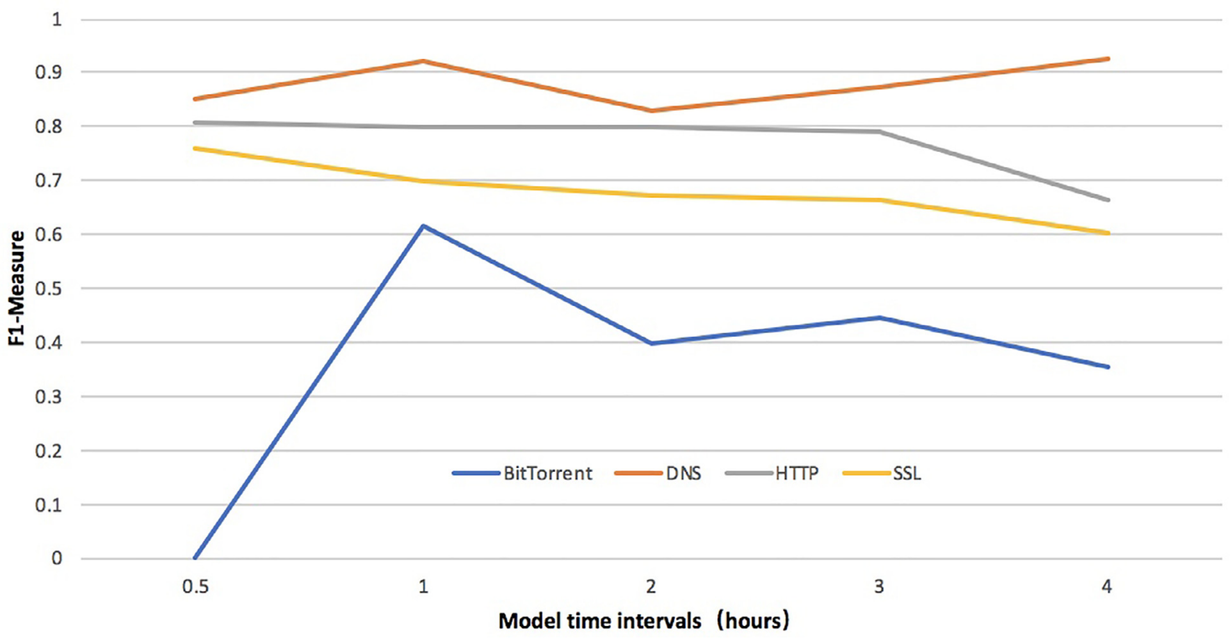

Fig. 7. $F1 - measure$ of models with different τ.

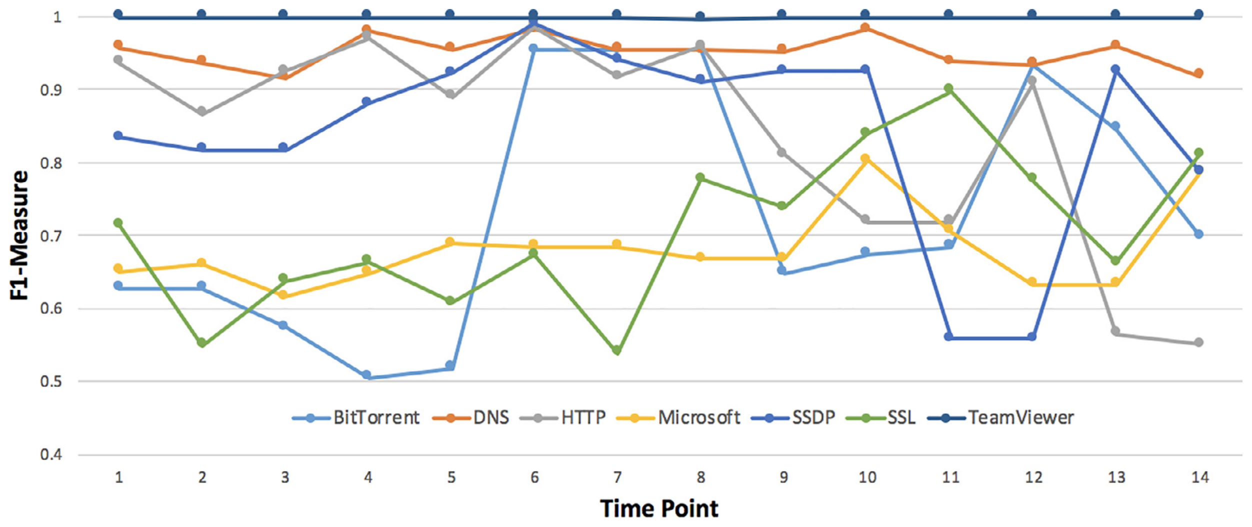

Fig. 8. $F1 - measure$ of the real-world stream experiment.

Table 3
Average metrics of the applications in real-world stream.

Application	Precision	Recall	F1-Measure
BitTorrent	0.829717	0.706831	0.7288
DNS	0.96026	0.945879	0.951728
HTTP	0.861711	0.851549	0.83769
Microsoft	0.58151	0.830641	0.681149
SSDP	0.814623	0.928269	0.842438
SSL	0.910802	0.617512	0.70649
TeamViewer	0.99998	0.99893	0.999452

Table 4
Distribution of different applications in the dynamic and imbalanced stream.

Application	Time0	Time1	Time2	Time3	Time3
DNS	1/6	1/5	1/4	1/5	1/6
360	1/6	2/15	1/12	2/15	1/6
BiTorrent	1/6	1/5	1/4	1/5	1/6
PPLive	1/6	2/15	1/12	2/15	1/6
SMTP	1/6	1/5	1/4	1/5	1/6
weibo	1/6	2/15	1/12	2/15	1/6

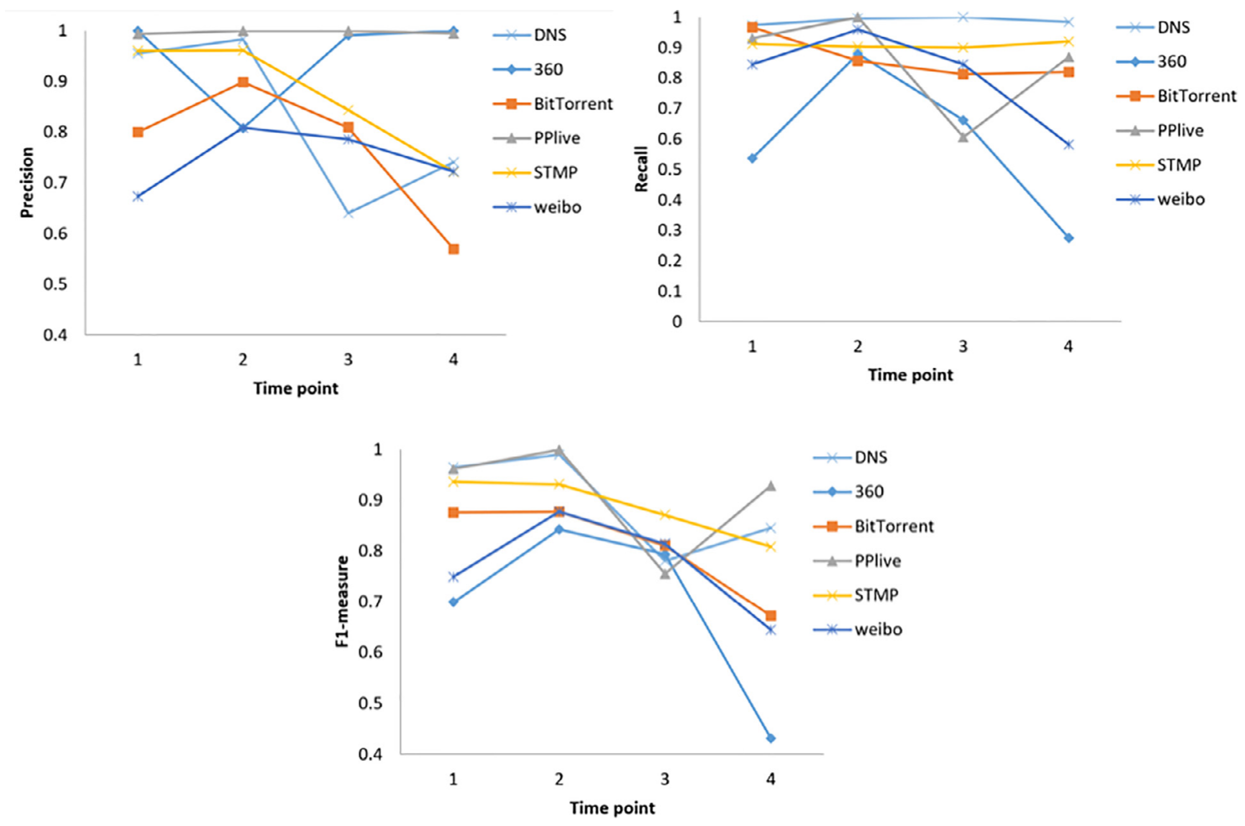

Fig. 9. *Precision* (a), *Recall* (b) and *F*1 − *measure* (c) of the dynamic and imbalance stream.

Table 3 shows the average metrics of all the time points for different applications. From this table, we can see the performance of DMTCS in detail. It can be found out that the average F1-measure for all protocols is higher than 70%. In all, DMTCS shows impressive performance in the real-world traffic stream, and it has enough accuracy to support network tasks.

5.9. Dynamic and imbalanced stream analysis

RQ6: For this question, we build a dynamic and imbalanced traffic stream for the experiment. In this stream, the numbers of packets from different applications change as time goes. The distribution of various applications at each time point in the stream of this experiment is listed in Table 4.

The numbers in each column of Table 4 represent the proportion of the specific application arrived at the corresponding time point. Time0 is the initializing time, time1 − 4 are different time points in our experiment.

Fig. 9 shows *Precision*, *Recall* and *F*1 − *measure* for each protocol at different time points in the dynamic stream. This time, we set the same parameter as those in Section 5.8. At first, it is evident that our system has a good performance for most applications in the dynamic and imbalanced environment. Except for the 360 at time point 4, all the classification results achieve F1-measure higher than 70%. Whats more, we can find out that Recall and Precision for 360 decrease rapidly from time point 2 to 3 and from time point 3 to 4, which may be that the training sets of the specific application are smaller

Table 5
Average metrics of the applications in dynamic stream.

Application	Precision	Recall	F1-Measure
DNS	0.829814	0.989167	0.895249
360	0.949498	0.58875	0.766697
BitTorrent	0.7692935	0.863959	0.808887
PPLive	0.996905	0.851875	0.911038
SMTP	0.87171	0.908958	0.886732
weibo	0.747487	0.807982	0.771535

Table 6
Comparisons results.

	PPLive			360			weibo			DNS		
Metrics	*P.*	*R.*	*F.*	*P.*	*R.*	*F.*	*P.*	*R.*	*F.*	*P.*	*R.*	*F.*
DMTCS	0.997	0.941	0.968	0.899	0.934	0.916	0.733	0.936	0.822	0.997	0.980	0.989
Se-SVM	0.999	0.799	0.888	0.595	0.93	0.726	0.635	0.961	0.764	0.951	1.0	0.975
Se-DT	1.0	0.951	0.975	0.922	0.677	0.774	0.642	0.801	0.713	0.99	0.996	0.993
Se-BN	0.951	0.975	0.952	0.656	0.929	0.769	0.609	0.742	0.669	0.991	0.998	0.995
nDPI	0.999	0.978	0.989	–	–	–	0.992	0.848	0.969	0.957	0.7677	0.8519

than the testing sets. In general, DMTCS can preserve high accuracy for most applications in the dynamic and imbalance stream.

Table 5 lists the average metrics of different time points in this experiment. It is clear from this table that DNS reaches the highest Recall of 98.91%, and PPlive achieves the highest Precision of 99.69%. The F1-measure of all protocols are higher than 75%. It can be concluded that our system has an outstanding performance in the dynamic and imbalanced environment.

5.10. Comparison

RQ7: We compare the classifying performance of DMTCS with Securitas and nDPI. Securitas proposed in 2016 [15] is one of the best solutions in the field of traffic classification. It is a payload-based traffic classification method with high accuracy. This method uses LDA to generate the signature information and adopts machine learning methods such as SVM, C4.5 Decision Tree, and Bayes network to do the traffic classification. nDPI is a model based on Deep Packet Inspection (DPI), and it is proposed in 2014 [9]. nDPI established a protocols library with application features from both the packet header and the packet payload, which indicates that a prior matching rule must be added to the library before classifying specific protocol.

In this experiment, we set the parameters of Securitas as described in [39], and set the parameters of DMTCS as the same as those in Section 5.6. Considering that Securitas is a model which can only distinguish one protocol from network traffic, we also let DMTCS recognize one target protocol from the same traffic. Although nDPI can classify hundreds of different protocols now, there are more applications that it cannot recognize. In this experiment, nDPI cannot identify 360 traffic. Meanwhile, we select four different target applications from our trace to compare the performance of different methods, i.e., PPlive, 360, weibo, and DNS. Note that these target protocols contain packets in both binary and text type.

Table 6 provides the comparisons results. In Table 6, Se-SVM, Se-DT, and Se-BN refer to Securitas using SVM, C4.5 Decision Tree, and Bayes network. Metrics P., R., and F. represent *Precision*, *Recall*, and $F1 - measure$.

As shown in Table 6, for PPlive, the *Precision* of DMTCS and Securitas are above 99%, and the Recall of those methods are higher than 94%. Securitas is a little better than DMTCS for this application. nDPI receives the best performance for PPlive with $F1 - measure$ 98.9%. While for 360, even though the *Precision* of Securitas is slightly higher than DMTCS, DMTCS is much better than Securitas in Recall. With regard to weibo, it is hard to figure out which is better by *Precision* and *Recall*. In the other metric, $F1 - measure$ of DMTCS is 82.2% while that of Securitas is 76.4%. nDPI achieves highest F1-measure as 96.9%. For DNS, Securitas methods and DMTCS show high F1-measure about 99%, and Securitas with Decision Tree is the best.

According to this experiment, it is clear that the classifying results of DMTCS are no worse than Securitas and nDPI. And in 360 protocols, DMTCS is superior to Securitas and nDPI. Whats more, comparing with Securitas, our system can deal with the multiple classification and the dynamic traffic stream. On the other hand, when comparing with nDPI, DMTCS does not need prior information of packets and can be trained for any protocols. In general, DMTCS may have more appealing aspects than Securitas and nDPI.

6. Conclusion

In this paper, we investigate dynamic classification for network traffic stream. To the best of our knowledge, little attention has been paid to this problem. We propose a classification system, named DMTCS, which can do the multi-classification in network traffic and adjust itself according to the dynamic traffic stream. Our method requires neither any prior information for the application nor the reconstruction of the network flow. And it can classify traffic from both TCP and UDP applications. Besides, it shows an impressive performance of classifying encrypted protocols, which means DMTCS can be applied to more areas. Our method is validated on the network traffic in the real world. The experiments show that DMTCS has excellent performance with F1-measure higher than 75% on our multiple classification experiments of two traffic traces. Our system shows advantages compared with the state-of-the-art methods.

In the future work, we intend to conduct some works to improve our system, such as establishing a new class for unknown protocols, or automatically finding the confidence of the classification result.

Acknowledgments

This work is supported by the NSFC projects (61773229, 61771273, 61202358), the National High-tech R&D Program of China (2015AA016102), the R&D Program of Shenzhen (JCYJ20160531174259309, JCYJ20170307153032483, JCYJ20160331184440545, JCYJ20170307153157440).

References

[1] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, K. Salamatian, Traffic classification on the fly, ACM SIGCOMM Comput. Commun. Rev. 36 (2) (2006) 23–26.
[2] L. Bernaille, R. Teixeira, K. Salamatian, Early application identification, in: Proceedings of the 2006 ACM CoNEXT conference, ACM, 2006, p. 6.
[3] D.M. Blei, A.Y. Ng, M.I. Jordan, Latent dirichlet allocation, J. Mach. Learn. Res. 3 (Jan) (2003) 993–1022.
[4] N. Borisov, D. Brumley, H.J. Wang, J. Dunagan, P. Joshi, C. Guo, Generic application-level protocol analyzer and its language., NDSS, 2007.
[5] J. Caballero, H. Yin, Z. Liang, D. Song, Polyglot: automatic extraction of protocol message format using dynamic binary analysis, in: Proceedings of the 14th ACM conference on Computer and communications security, ACM, 2007, pp. 317–329.
[6] Z. Chen, Q. Yan, H. Han, S. Wang, L. Peng, L. Wang, B. Yang, Machine learning based mobile malware detection using highly imbalanced network traffic, Inf. Sci. 433 (2018) 346–364.
[7] P.M. Comparetti, G. Wondracek, C. Kruegel, E. Kirda, Prospex: protocol specification extraction, in: Security and Privacy, 2009 30th IEEE Symposium on, IEEE, 2009, pp. 110–125.
[8] W. Cui, M. Peinado, K. Chen, H.J. Wang, L. Irun-Briz, Tupni: automatic reverse engineering of input formats, in: Proceedings of the 15th ACM conference on Computer and communications security, ACM, 2008, pp. 391–402.
[9] L. Deri, M. Martinelli, T. Bujlow, A. Cardigliano, ndpi: open-source high-speed deep packet inspection, in: Wireless Communications and Mobile Computing Conference (IWCMC), 2014 International, IEEE, 2014, pp. 617–622.
[10] P. Ducange, G. Mannarà, F. Marcelloni, R. Pecori, M. Vecchio, A novel approach for internet traffic classification based on multi-objective evolutionary fuzzy classifiers, in: 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE, 2017, pp. 1–6.
[11] J. Erman, M. Arlitt, A. Mahanti, Traffic classification using clustering algorithms, in: Proceedings of the 2006 SIGCOMM workshop on Mining network data, ACM, 2006, pp. 281–286.
[12] A. Finamore, M. Mellia, M. Meo, D. Rossi, Kiss: stochastic packet inspection, in: International Workshop on Traffic Monitoring and Analysis, Springer, 2009, pp. 117–125.
[13] A. Finamore, M. Mellia, M. Meo, D. Rossi, Kiss: stochastic packet inspection classifier for udp traffic, IEEE/ACM Trans. Netw. (TON) 18 (5) (2010) 1505–1515.
[14] P. Haffner, S. Sen, O. Spatscheck, D. Wang, Acas: automated construction of application signatures, in: Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data, ACM, 2005, pp. 197–202.
[15] N.-F. Huang, G.-Y. Jai, H.-C. Chao, Y.-J. Tzang, H.-Y. Chang, Application traffic classification at the early stage by characterizing application rounds, Inf. Sci. 232 (2013) 130–142.
[16] M. Iliofotou, M. Faloutsos, M. Mitzenmacher, Exploiting dynamicity in graph-based traffic analysis: techniques and applications, in: Proceedings of the 5th international conference on Emerging networking experiments and technologies, ACM, 2009, pp. 241–252.
[17] R. Jin, A.G. Hauptmann, C.X. Zhai, Language model for information retrieval, in: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 2002, pp. 42–48.
[18] J. Kannan, J. Jung, V. Paxson, C.E. Koksal, Semi-automated discovery of application session structure, in: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement, ACM, 2006, pp. 119–132.
[19] T. Karagiannis, K. Papagiannaki, M. Faloutsos, Blinc: multilevel traffic classification in the dark, in: ACM SIGCOMM Computer Communication Review, 35, ACM, 2005, pp. 229–240.
[20] H. Kim, K.C. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, K. Lee, Internet traffic classification demystified: myths, caveats, and the best practices, in: Proceedings of the 2008 ACM CoNEXT conference, ACM, 2008, p. 11.
[21] G. La Mantia, D. Rossi, A. Finamore, M. Mellia, M. Meo, Stochastic packet inspection for tcp traffic, in: Communications (ICC), 2010 IEEE International Conference on, IEEE, 2010, pp. 1–6.
[22] J. Li, X. Ma, J. Zhang, J. Tao, P. Wang, X. Guan, Mining repeating pattern in packet arrivals: metrics, models, and applications, Inf. Sci. 408 (2017) 1–22.
[23] Z. Li, G. Xia, H. Gao, Y. Tang, Y. Chen, B. Liu, J. Jiang, Y. Lv, Netshield: massive semantics-based vulnerability signature matching for high-speed networks, in: ACM SIGCOMM Computer Communication Review, 40, ACM, 2010, pp. 279–290.
[24] S. Liang, E. Yilmaz, E. Kanoulas, Dynamic clustering of streaming short documents, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 995–1004.
[25] J. Lim, T. Reps, B. Liblit, Extracting output formats from executables, in: Reverse Engineering, 2006. WCRE'06. 13th Working Conference on, IEEE, 2006, pp. 167–178.
[26] Y.-s. Lim, H.-c. Kim, J. Jeong, C.-k. Kim, T.T. Kwon, Y. Choi, Internet traffic classification demystified: on the sources of the discriminative power, in: Proceedings of the 6th International COnference, ACM, 2010, p. 9.
[27] M. Lotfollahi, R. Shirali, M.J. Siavoshani, M. Saberian, Deep packet: a novel approach for encrypted traffic classification using deep learning, arXiv preprint arXiv:1709.02656 (2017).
[28] J. Ma, K. Levchenko, C. Kreibich, S. Savage, G.M. Voelker, Unexpected means of protocol inference, in: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement, ACM, 2006, pp. 313–326.
[29] C. Meiners, E. Norige, A.X. Liu, E. Torng, Flowsifter: a counting automata approach to layer 7 field extraction for deep flow inspection, in: INFOCOM, 2012 Proceedings IEEE, IEEE, 2012, pp. 1746–1754.
[30] T.T. Nguyen, G. Armitage, A survey of techniques for internet traffic classification using machine learning, IEEE Commun. Surv. Tut. 10 (4) (2008) 56–76.
[31] R. Pang, V. Paxson, R. Sommer, L. Peterson, binpac: a yacc for writing application protocol parsers, in: Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement, ACM, 2006, pp. 289–300.

[32] C.S. Sastry, S. Rawat, A.K. Pujari, V.P. Gulati, Network traffic analysis using singular value decomposition and multiscale transforms, Inf. Sci. 177 (23) (2007) 5275–5291.
[33] J. St Sauver, A look at the unidentified half of netflow (with an additional tutorial on how to use the internet2 netflow data archives)(2008).
[34] A. Tongaonkar, R. Torres, M. Iliofotou, R. Keralapura, A. Nucci, Towards self adaptive network traffic classification, Comput. Commun. 56 (2015) 35–46.
[35] W. Wang, M. Zhu, J. Wang, X. Zeng, Z. Yang, End-to-end encrypted traffic classification with one-dimensional convolution neural networks, in: Intelligence and Security Informatics (ISI), 2017 IEEE International Conference on, IEEE, 2017, pp. 43–48.
[36] W. Wang, M. Zhu, X. Zeng, X. Ye, Y. Sheng, Malware traffic classification using convolutional neural network for representation learning, in: Information Networking (ICOIN), 2017 International Conference on, IEEE, 2017, pp. 712–717.
[37] Y. Wu, G. Min, K. Li, B. Javadi, Modeling and analysis of communication networks in multicluster systems under spatio-temporal bursty traffic, IEEE Trans. Parallel Distrib. Syst. 23 (5) (2012) 902–912.
[38] Y. Xiang, W. Zhou, M. Guo, Flexible deterministic packet marking: an ip traceback system to find the real source of attacks, IEEE Trans. Parallel Distrib. Syst. 20 (4) (2009) 567–580.
[39] X. Yun, Y. Wang, Y. Zhang, Y. Zhou, A semantics-aware approach to the automated network protocol identification, IEEE/ACM Trans. Netw. (TON) 24 (1) (2016) 583–595.
[40] J. Zhang, C. Chen, Y. Xiang, W. Zhou, A.V. Vasilakos, An effective network traffic classification method with unknown flow detection, IEEE Trans. Netw. Serv. Manage. 10 (2) (2013) 133–147.
[41] J. Zhang, Y. Xiang, Y. Wang, W. Zhou, Y. Xiang, Y. Guan, Network traffic classification using correlation information, IEEE Trans. Parallel Distrib. Syst. 24 (1) (2013) 104–117.
[42] J. Zhang, Y. Xiang, W. Zhou, Y. Wang, Unsupervised traffic classification using flow statistical properties and ip packet payload, J. Comput. Syst. Sci. 79 (5) (2013) 573–585.