Harvesting Unique *Characteristics in Packet Sequences* for Effective Application Classification

Zhenlong Yuan
Department of Automation
Tsinghua University
Beijing, China
Email: yuanzl11@mails.tsinghua.edu.cn

Yibo Xue
Tsinghua National Lab for
Information Science and Technology
Beijing, China
Email: yiboxue@tsinghua.edu.cn

Yingfei Dong
Department of Electrical Engineering
University of Hawaii, Honolulu
HI 96822, USA
Email: yingfei@hawaii.edu

***Abstract*—Network traffic classification is critical to both network management and security. Identifying application traffic at the flow level with signature matching has been widely used as the most efficient method due to its reliability and robustness. However, due to the increasing number of applications and their frequent updates, we have to constantly regenerate application signatures, which is both resource intensive and time consuming. To address this issue, we propose to explore the unique *characteristics in packet sequences* and discovered two types of packet sequence signatures. We introduce our design and implementation of an automated packet-sequence signature construction (APSC) system, based on association rule mining and data clustering technologies. This system can not only automatically generate traditional signatures from individual packet payloads but also construct new packet sequence signatures based on payloads or features from packet sequences, even for encrypted flows. To the best of our knowledge, this is the first practical and efficient system that supports automated packet sequence signature construction. Our experimental results show that the proposed system can automatically construct high quality signatures for a variety of application with limited overhead.**

Keywords*—Traffic classification, network management, packet sequence signature, automated signature construction.

I. Introduction

Associating traffic with its corresponding applications has a great influence on network management (e.g., Quality of Service) and security (e.g., network intrusion detection systems). It allows network administrators to know exactly which applications are running and then take timely actions, such as assigning higher flow priority or limiting flow transmission rates. If undesired or malicious traffic is detected, administrators can filter them to protect their networks.

Because of such practical use, various traffic classification techniques have been developed, such as early port-based techniques, subsequently signature-based techniques, and later statistics-based techniques [1]–[3]. Among these schemes, signature-based techniques are usually considered the most reliable and efficient due to the uniqueness and robustness of signatures, especially in industry for classifying real-time network traffic. In practice, due to the increasing number of applications and their frequent updates, application traffic may change frequently and significantly change, and the effectiveness of static signatures become limited. Consequently, we have to repeatedly regenerate application signatures. Currently, application signatures are generally derived by manually or semi-manually analyzing large amounts of traffic over long periods, which is both resource intensive and time consuming.

Although automatically generating signatures has been studied in many projects [4]–[7], researchers mostly focused on finding signatures in individual packet payload. Very little efforts have been done in exploiting the *characteristics in packet sequences*. Based on our experimental investigation, we have found that such sequence signatures of application flows can help us improve the accuracy and efficiency of classifying applications.

In this paper, we first introduce two types of packet sequence signatures that we have discovered, and then present our design and implementation of an automated packet-sequence signature construction (APSC) system, based on association rule mining and data clustering technologies. Our experimental results show that APSC can generate high-quality traditional signatures and new packet sequence signatures for many applications. To the best of our knowledge, this is the first practical and efficient work on automatically generating application signature based on packet sequences, instead of solely based on individual packets. The main contributions of this paper include:

1) *Identify unique characteristics in packet sequences*. From our experimental observations and analysis, we discover payload-based and feature-based signatures based on packet sequences. Our idea of feature-based packet sequence signature is derived from the common Machine Learning (ML) C4.5 decision tree algorithm.
2) *Design an automated signature construction system*. We have designed and implemented APSC based on association rules mining and divisive hierarchical clustering. Both traditional signatures and the newly proposed packet sequence signatures can be constructed automatically with limited overhead.
3) *Develop efficient schemes to apply the new packet sequence signatures*. We have developed an efficient regex-engine method to apply packet sequence signatures on real-time traffic with practical overhead.

The reminder of this paper is organized as follows. We first briefly review the related work in Sec.II, and then present our key observations in Sec.III. We present the design and implementation of the proposed system in Sec.IV, and evaluate the system in Sec.V. We conclude this paper in Sec.VI.

978-1-4799-0895-0/13/$31.00 ©2013 IEEE

II. Related Work

A. Network Traffic Classification

Network traffic classification has been evolving with the development of Internet. Port-based techniques are the earliest methods to classify traffic based on well-known port numbers [8]. However, such techniques have become ineffective due to the widely use of random port numbers in applications (e.g., Bittorrent). To address this issue, payload signature-based techniques are developed to identify application traffic by matching packet payloads with known string patterns and regular expressions [9]–[12]. Because of the reliability and robustness of these signatures, these techniques are widely adopted in many fields, such as L7-filter for application layer packet classification [13] and Snort for network intrusion detection [14]. In industry, signature-based techniques are also considered as the most efficient, especially when dealing with real-time traffic. We have to acknowledge that signature-based solutions are still one of the most indispensable elements for traffic classification.

Since the seminal paper by McGregor et al. [15] on applying ML techniques for traffic classification with the Expectation Maximization algorithm, various statistics-based techniques have been developed in the past decade, exploring flows statistics instead of only packet payload patterns, based on various ML algorithms such as Bayesian [16], [17], Support Vector Machine [18], [19] and C4.5 [20], [21]. These techniques generally assume that flows generated by different applications have unique statistical characteristics. However, when we apply these techniques to deal with high traffic loads (e.g., on backbone networks), where million of flows to be classified in a second, the precision and efficiency of such methods are seriously decreased, proved by our previous experiments [22], [23]. They are unable to meet the real-time requirements and process heavy traffic.

B. Automated Signature Generation

Because manually deriving signatures from large volumes of application traffic is resource intensive and time consuming, automatically generating signatures has become an urgent task. Haffner et al. [4] applied three statistical ML algorithms to automatically identify signatures for a range of applications. Park et al. [5] focused on the string patterns in packet payloads and proposed the *LASER* algorithm. Ye et al. [6] proposed the *AutoSig* system which extracts multiple common substring sequences from sample flows as application signatures. Recently, Wang et al. [7] focused on generating signatures with a subset of standard syntax rules and proposed an automatic signature generation system. For comparison in later sections, we name it *AppID* for short. The biggest difference between their methods and ours is that we not only support traditional signatures generation automatically but also build *new packet sequence signatures* automatically for the first time.

Because of the fact that a 5-tuple flow must be generated by an application, we do not have to identify application flows with only a single packet payload. Just like ML-based techniques that use the characteristics of multiple packets as features for model building, we use *a sequence of packet payloads* for application identification. For instance, although one byte in a single packet may not be treated as a signature, a pattern constructed by multiple bytes in a packet sequence of an application is likely to be an effective signature. We have confirmed this with our experiments.

III. Observations and Analysis

In this section, we present several crucial observations that motivate us to build payload-based and feature-based signatures based on packet sequences.

A. Payload-based Packet Sequence Signatures

In the past decades, researchers mostly focused on finding effective signatures in individual packet payloads due to various reasons. However, an application normally issues a sequence of packets. We have seen almost no efforts in exploiting such characteristics in packet sequences. Based on our experiments, we found that such a method is feasible to classify applications. Therefore, as ML-based techniques that use flow-level characteristics such as the packets sizes or packet arrival intervals, we can classify network traffic based on *characteristic in packet sequences.*

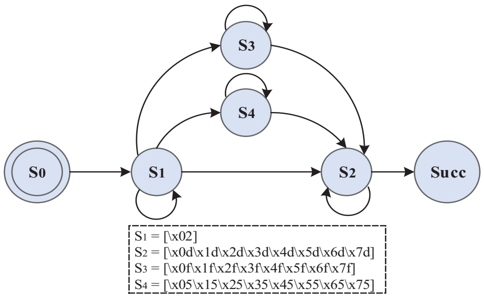
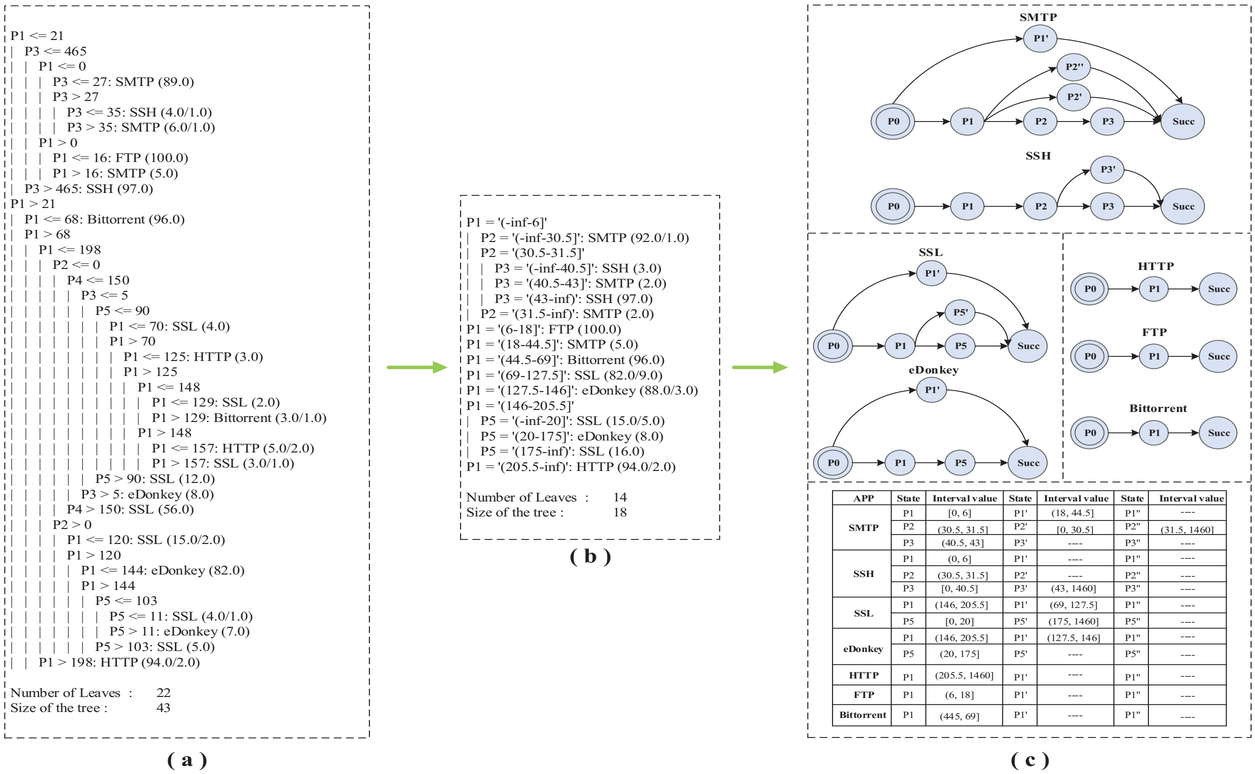

S0
S1
S2
S3
S4
Succ
S1 = [\x02]
S2 = [\x0d\x1d\x2d\x3d\x4d\x5d\x6d\x7d]
S3 = [\x0f\x1f\x2f\x3f\x4f\x5f\x6f\x7f]
S4 = [\x05\x15\x25\x35\x45\x55\x65\x75]

Fig. 1: Packet Sequence Signature of Skype Media Flows.

Fig.1 shows the sequence signature of Skype media flows we discovered. The arrow represents a sequence transition between two states, the state S_0 represents for the beginning state, and the state $S_n, n = 1, 2, 3, 4$ with a simple regular expression represents the third byte hexadecimal value of a packet payload in a Skype flow. Moreover, with the help of this new sequence signature, the media traffic that Skype generates can be entirely identified, including voice-calls, skypeOut, file transfer, and video conferencing. Although various methods have been used to classify Skype traffic, to the best of our knowledge, it was the first time to discover such a sequence signature for Skype. In the meantime, based on our experiments, this is probably the easiest way to identify Skype traffic with a high accuracy. We will present the detailed evaluation in Sec.V.

A sequence signature reflects the process of interactive communications for an application. In general, an application has a negotiation phase before a data transmission phase, *even if it uses encryption*. We can exploit the sequence signature in the negotiation phase to achieve early identification of such flows.



```
P1 <= 21
|   P3 <= 465
|   |   P1 <= 0
|   |   |   P3 <= 27: SMTP (89.0)
|   |   |   P3 > 27
|   |   |   |   P3 <= 35: SSH (4.0/1.0)
|   |   |   |   P3 > 35: SMTP (6.0/1.0)
|   |   P1 > 0
|   |   |   P1 <= 16: FTP (100.0)
|   |   |   P1 > 16: SMTP (5.0)
|   P3 > 465: SSH (97.0)
P1 > 21
|   P1 <= 68: Bittorrent (96.0)
|   P1 > 68
|   |   P1 <= 198
|   |   |   P2 <= 0
|   |   |   |   P4 <= 150
|   |   |   |   |   P3 <= 5
|   |   |   |   |   |   P5 <= 90
|   |   |   |   |   |   |   P1 <= 70: SSL (4.0)
|   |   |   |   |   |   |   P1 > 70
|   |   |   |   |   |   |   |   P1 <= 125: HTTP (3.0)
|   |   |   |   |   |   |   |   P1 > 125
|   |   |   |   |   |   |   |   |   P1 <= 148
|   |   |   |   |   |   |   |   |   |   P1 <= 129: SSL (2.0)
|   |   |   |   |   |   |   |   |   |   P1 > 129: Bittorrent (3.0/1.0)
|   |   |   |   |   |   |   |   |   P1 > 148
|   |   |   |   |   |   |   |   |   |   P1 <= 157: HTTP (5.0/2.0)
|   |   |   |   |   |   |   |   |   |   P1 > 157: SSL (3.0/1.0)
|   |   |   |   |   |   P5 > 90: SSL (12.0)
|   |   |   |   |   P3 > 5: eDonkey (8.0)
|   |   |   |   P4 > 150: SSL (56.0)
|   |   |   P2 > 0
|   |   |   |   P1 <= 120: SSL (15.0/2.0)
|   |   |   |   P1 > 120
|   |   |   |   |   P1 <= 144: eDonkey (82.0)
|   |   |   |   |   P1 > 144
|   |   |   |   |   |   P5 <= 103
|   |   |   |   |   |   |   P5 <= 11: SSL (4.0/1.0)
|   |   |   |   |   |   |   P5 > 11: eDonkey (7.0)
|   |   |   |   |   |   P5 > 103: SSL (5.0)
|   |   P1 > 198: HTTP (94.0/2.0)

Number of Leaves  :     22
Size of the tree :      43
```

(a)

```
P1 = '(-inf-6]'
|   P2 = '(-inf-30.5]': SMTP (92.0/1.0)
|   P2 = '(30.5-31.5]'
|   |   P3 = '(-inf-40.5]': SSH (3.0)
|   |   P3 = '(40.5-43]': SMTP (2.0)
|   |   P3 = '(43-inf)': SSH (97.0)
|   P2 = '(31.5-inf)': SMTP (2.0)
P1 = '(6-18]': FTP (100.0)
P1 = '(18-44.5]': SMTP (5.0)
P1 = '(44.5-69]': Bittorrent (96.0)
P1 = '(69-127.5]': SSL (82.0/9.0)
P1 = '(127.5-146]': eDonkey (88.0/3.0)
P1 = '(146-205.5]'
|   P5 = '(-inf-20]': SSL (15.0/5.0)
|   P5 = '(20-175]': eDonkey (8.0)
|   P5 = '(175-inf)': SSL (16.0)
P1 = '(205.5-inf)': HTTP (94.0/2.0)

Number of Leaves  :     14
Size of the tree :      18
```

(b)

APP	State	Interval value	State	Interval value	State	Interval value
SMTP	P1	[0, 6]	P1'	(18, 44.5]	P1"	----
	P2	(30.5, 31.5]	P2'	[0, 30.5]	P2"	(31.5, 1460]
	P3	(40.5, 43]	P3'	----	P3"	----
SSH	P1	(0, 6]	P1'	----	P1"	----
	P2	(30.5, 31.5]	P2'	----	P2"	----
	P3	[0, 40.5]	P3'	(43, 1460]	P3"	----
SSL	P1	(146, 205.5]	P1'	(69, 127.5]	P1"	----
	P5	[0, 20]	P5'	(175, 1460]	P5"	----
eDonkey	P1	(146, 205.5]	P1'	(127.5, 146]	P1"	----
	P5	(20, 175]	P5'	----	P5"	----
HTTP	P1	(205.5, 1460]	P1'	----	P1"	----
FTP	P1	(6, 18]	P1'	----	P1"	----
Bittorrent	P1	(445, 69]	P1'	----	P1"	----

(c)

Fig. 2: Packet Sequence Signature with C4.5 Decision Tree.

B. Feature-based Packet Sequence Signatures

In recent years, statistical-feature-based methods are widely studied for traffic classification. The rationale is that traffic generated by different types of applications exhibits distinct characteristics such as packet size, time interval, etc. Packets sizes are generally regarded as the most effective features used for ML-based techniques. Based on our observations and analysis, the essential reason why the use of packet size works very well in ML-based classification is due to the different designs of various application protocols. In addition, researchers have found that the C4.5 algorithm exhibits the best performance under many circumstances in accuracy and speed [1], [21]. By selecting the first five packets sizes as features, Fig.2(a) shows the C4.5 training model generated on the Weka platform [24]. We can see that the classification process is actually identifying the intervals that packets sizes fall into. To make it more intuitive, although C4.5 itself has involved an entropy-based discretization technique to deal with numeric and continuous features, we employ an entropy-based minimum-description-length discretization technique to deal with features before inputting them to the C4.5 algorithm. Our experimental results show that both discrete and continuous input data show similar classification accuracies.

As Fig.2(b) shows, the classification issue is indeed becoming an interval judgment problem. Hence we can infer that it is the different interval distribution of packet sizes that substantially contributes to the high accuracy in packet-size-based traffic classification. Furthermore, comparing with Fig.2(a), in Fig.2(b) (i) both the number of leaves and the size of C4.5 decision tree are decreased, and (ii) the tree breadth is increased while the tree depth is decreased. The reason for the changes from Fig.2(a) to Fig.2(b) is due to the input discrete features, not the traditional continuous features. Furthermore, due to the usage of discrete features, the C4.5 decision tree does not need to judge the same packet size many times as before. Note that the depth of C4.5 decision tree in Fig.2(b) is less than that in Fig.2(a). As a result, if we employ discrete features for training a C4.5 decision tree, the classification speed must be improved effectively. In fact, the above practice increases the tree breadth in exchange for a smaller depth.

In the past, without considering the statistical features in packet sequences for online real-time traffic classification, the classification techniques have to wait for collecting all the necessary features and then input the feature values to ML models for identification. However, the packet sequence number actually stands for a time sequence in a flow. So, we can construct a deterministic finite automaton (DFA) for each application flow based on the decision tree shown in Fig.2(b) by taking the packet sequence number as a time sequence signal. Moreover, all the DFAs derived from Fig.2(b) can merge into to a composite DFA. Fig.2(c) shows the feature-based DFAs, where P_n, P'_n, or P''_n represents the matching condition for the n-th packet in a flow with an interval value, $n = 1, 2, 3, 4, 5$. In most matching conditions, the interval value is relatively large because a large one is enough to classify the limited number of applications. Nevertheless, according to our observations, a great number of network applications indeed have fixed small intervals or even determined values for some packets sizes. Considering that the intervals or determined values are just like signatures for application identification, we use them as the feature-based packet sequence signatures for traffic classification. The detailed experimental evaluation will be presented in Sec.V.

To the best of our knowledge, this is the first work to use feature-based packet sequence signatures for traffic classification. In this method, instead of wasting time and memory to gather a large number of packets sizes and then inputting them to the ML model, we just need to maintain one status number of the composite DFA for each flow. Therefore, comparing with traditional ML techniques, the proposed packet sequence feature-based signature can not only improve the classification speed but also reduce memory consumption considerably.

IV. APSC SYSTEM

In this section, we introduce the proposed APSC system which can automatically construct packet sequence signatures for various network applications. It is based on association rules mining technology and data clustering technology. APSC is motivated by our observations discussed in the above section, and it can build both traditional single-payload signatures and our new packet sequence signatures. In the following, we discuss the key requirements for automated signature construction system, introduce the related terminology and definitions, and then present the system architecture in detail.

A. System Goals

To classify traffic with signatures in a real-time for high-speed networks, we need an automated signature construction system that meets the following requirements.

High-accuracy. For large-scale traffic classification, we need unique and robust signatures for high accuracy and low misclassification rate. For example, for a backbone router at 230 Gbytes per second [25], 1% of classification error rate will result in incorrect classification of 2.30 Gbytes per second. It may heavily affect network administrators' actions and seriously degrade user experience. Therefore, high-accuracy is critical for traffic classification.

Low-overhead. For identifying applications in real-time on high speed links, low processing overhead and low memory consumption are essential for signature matching. In addition, the more network traffic we analyze, the more accurate the signatures we can obtain. So the ability to process large amounts of traffic with low overhead is also an indispensable requirement for automated signature construction system, especially when we are facing constant application updates. Therefore, we need both efficient signature matching and efficient signature generating.

Early Identification. In general, accurate signatures are needed for quick responses. For example, an ISP may like to change the service class of a flow to provide better QoS as soon as possible; a network administrator may take immediate actions against undesired traffic. It is vital to make a decision at the early stage of a flow. Otherwise, the QoS may be violated and the undesired traffic may cause serious damages. Such a requirement of early identification raises new challenges to automated signature generation.

Long-term Effectiveness. With the increasing number of network applications and their frequent updates, the traffic pattern of applications may change frequently. Therefore, signatures generated by automated signature construction must be robust enough to adapt to these changes. In other words, to avoid duplicate efforts, signatures must have long-term effectiveness.

Unidirectional Identification. On current networks, we usually capture only unidirectional flow-level information at a measurement point, due to the prevalence of asymmetric routing. As a result, only one direction of a bidirectional communication can be used for signature matching to identify application traffic. For this reason, an automated signature construction system should find effective signatures for each direction separately.

B. Terminology and Definitions

Data mining methods extract knowledge from a large amount of data to help us identify hidden trends and patterns [26]. Association rule mining technology is a well-evaluated method for discovering interesting relations between variables in large databases. In this project, we consider individual elements such as byte values in packet payloads or feature values in a 5-tuple flow descriptors as targeted variables, and consider signatures as mining association rules. In the association rule method, (i) a ***transaction database*** consists of records representing ***transactions***, while a transaction is a list of ***items***. (ii) an ***itemset*** is a set of items that appear together in a transaction data set. (iii) A ***frequent itemset*** represents for an itemset that occur frequently in large transaction datasets. (iv) a ***support*** is an objective measure of itemset frequency, representing the percentage of transactions from a transaction database that a given itemset includes. (v) a ***confidence*** is also objective measure for association rules, which assesses the degree of certainty of the detected association.

Let $I = \{I_1, I_2, ..., I_m\}$ be a set of items. Let D, the task-relevant data, be a set of database transactions, where each transaction T is a set of items such that $T \subseteq I$. Each transaction is associated with an identifier, called *TID*. Let A be a set of items. Transaction T is said to contain A if and only if $A \subseteq T$. An association rule is an implication of the form $A \Rightarrow B$, where $A \subset I, B \subset I$, and $A \cap B = \phi$. The rule $A \Rightarrow B$ holds in the transaction set D with *support* s, where s is the percentage of transactions in D that contain $A \cup B$, with a probability $P(A \cup B)$. The rule $A \Rightarrow B$ has *confidence* c in the transaction set D, where c is the percentage of transactions in D containing A that contain B, with a conditional probability $P(B|A)$. That is,

$$support(A \Rightarrow B) = P(A \cup B) \tag{1}$$

$$confidence(A \Rightarrow B) = P(B|A) \tag{2}$$

An itemset that contains k items is called k**-itemset**. For example, the set $\{I_1, I_2\}$ is a 2-itemset. The occurrence frequency of an itemset is the number of transactions that

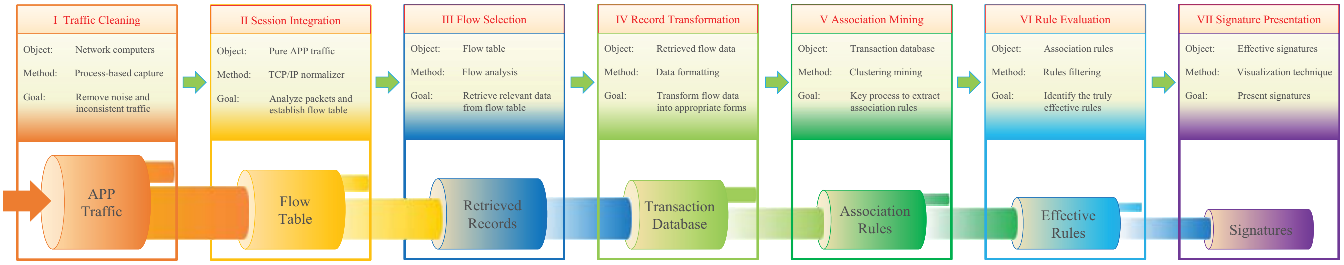

I Traffic Cleaning
Object: Network computers
Method: Process-based capture
Goal: Remove noise and inconsistent traffic
II Session Integration
Object: Pure APP traffic
Method: TCP/IP normalizer
Goal: Analyze packets and establish flow table
III Flow Selection
Object: Flow table
Method: Flow analysis
Goal: Retrieve relevant data from flow table
IV Record Transformation
Object: Retrieved flow data
Method: Data formatting
Goal: Transform flow data into appropriate forms
V Association Mining
Object: Transaction database
Method: Clustering mining
Goal: Key process to extract association rules
VI Rule Evaluation
Object: Association rules
Method: Rules filtering
Goal: Identify the truly effective rules
VII Signature Presentation
Object: Effective signatures
Method: Visualization technique
Goal: Present signatures
APP Traffic
Flow Table
Retrieved Records
Transaction Database
Association Rules
Effective Rules
Signatures

Fig. 3: Overview of APSC System.

contain the itemset. If the relative support of an itemset I satisfies a prespecified minimum support threshold (min_sup), then I is a **frequent itemset**. On the contrary, if an itemset I' owns the maximal frequency in a cluster, then I' is of the **maximal support** (max_sup). In this paper, we denote support and confidence values between 0% and 100%, rather than 0 to 1.0. From Equation (2), we have

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support(A \cup B)}{support(A)} \qquad (3)$$

C. System Overview

The proposed system is designed on the basis of Knowledge Discovery in Database (KDD), and Fig.3 shows the system overview. Similar as the main steps in KDD [26] [27], there are seven processing steps in our system implementation. (1) *Traffic cleaning*. The goal of this step is to obtain the pure traffic of an application. A simple way to achieve this is to capture traffic at the gateway with all the hosts in a network running the same application. A better way (regarded as the best) is to capture traffic on different computers based on the application process with the Commview [28] tool and then mix them together. (2) *Session integration* tracks and normalizes sessions with packet analysis to establish a flow table. (3) *Flow selection*. After establishing the flow table, flows relevant to the analysis task are retrieved from the flow table for flow analysis. (4) *Record transformation*. In this stage, the retrieved flow data are transformed into transaction databases through our distinct level data formatting techniques. The detailed data formatting techniques appropriated for association rules mining will be introduced later this section. (5) *Association mining* is the key and essential process of our system where association rules mining technology and clustering technology are applied to extract association rules. (6) *Rule evaluation* is used for identifying the truly effective association rules representing signatures based on rule filtering techniques. (7) *Signature presentation*. For better illustration, we use a visualization method to present the effective association rules and signatures.

For our investigation, we emphasis on the *association mining* part. Techniques for distinct level data formatting and the process of *rule evaluation* are also described. The detailed system architecture is shown in Fig.4.

D. Preprocessing Module

Considering traditional signatures and the proposed two new packet sequence signatures, we divide the data formatting process into two distinct levels, i.e., packet-level data formatting for payload-based signature generation and flow-level data formatting for feature-based signature generation. In particular, for feature-based signatures, we take packet sizes as features.

1) Packet-level Data Formatting: In order to extract payload-based signatures, we consider one packet as a *transaction* and each byte of the packet as an *item*. Then a *transaction database* consists of packets that belong to one specific application. More specifically, we combine the value of a byte with its offset in a payload to represent an item. Note that as the length of a common packet is distributed from 0 to 1460, the number of items in different transactions changes correspondingly.

2) Flow-level Data Formatting: Similarly, for extracting feature-based signatures, we consider one flow as a *transaction* and each packet of the flow as an *item*. Then the *transaction database* consists of flows that belong to one specific application. More precisely, we combine the size of a packet with its sequence number in a flow to represent an item. Similar to the packet-level data formatting, as the size of flows are not fixed, the number of items in different transactions changes dynamically.

E. Association Rules Mining (ARM) Module

The association rules mining module process both levels of data formatting in the same manner. In this module, we combine a divisive hierarchical clustering method with the association rules mining technology, i.e., we select the association rules as the clustering criterion for divisive hierarchical clustering. The divisive approach, also called the top-down approach, starts with all objects in the same cluster and processes them by multiple iterations. In each iteration, a cluster is split into smaller clusters, until eventually each object is in one cluster, or until a termination condition meets. In this module, we first define the clustering criteria for clustering, and then introduce the divisive hierarchical clustering method for signature construction in detail.

Clustering Criteria. When a transaction database of an application is sufficiently large, its hidden signatures will consist of at least one or more frequent items in the transaction database. Therefore, we select the frequent items with *max_sup* in a cluster as the clustering criteria.

Divisive Hierarchical Clustering. Fig.5 depicts the divisive hierarchical clustering process. Initially, all the transactions are classified into the same cluster. Then it begins to find the

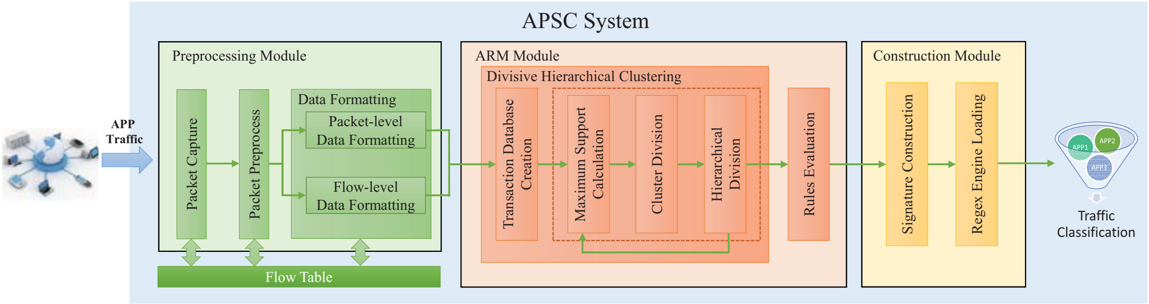


Fig. 4: Architecture of APSC System.

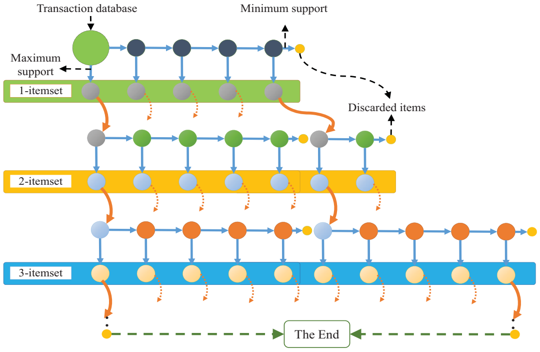


Fig. 5: Divisive Hierarchical Clustering.

maximal frequent item with *max_sup* among all items. The frequent item having *max_sup* must be 1-itemset because any k-itemset consists of 1-itemsets. Transactions with the maximal frequent item can be extracted from the original cluster and forms a new cluster. At the same time, the other transactions form a new cluster to be divided according to new thresholds. Eventually, we divide all the transactions into diverse clusters until there are no frequent item. As a cluster is usually divided into sub-clusters, we name the original cluster as the *parent node* of the sub-clusters and the sub-clusters as the *child nodes* of the original cluster.

The sub-clusters identified in the first iteration are considered at the first layer of divisive hierarchical clustering. We then iteratively divide these clusters into children clusters at a lower layer. Note that the items that have been used as the clustering criteria in the parent nodes will not be used as the clustering criteria in the child nodes. Therefore, in each layer, the clustering criteria is always based on 1-itemset. The iterative processing will not stop until a desired number of clusters is obtained or the *max_sup* of each cluster is within a pre-set threshold. From Fig.5, we can see that, compared with parent nodes, the maximal frequent itemset of child nodes adds one more item, e.g., the maximal frequent itemset of clusters in the first layer is 1-itemset, while that in the second layer is 2-itemset. Clearly, after the divisive hierarchical clustering process, a tree is produced with each node being a cluster.

Rules Evaluation. The proposed system takes signatures as association rules and tries to extract effective signatures by association rule mining. However, not all association rules extracted from traffic will be effective, e.g., the payload-based signature “Get /” is likely to occur in various kinds of application traffic, and the feature-based signature “0x5B4” stands for the largest packet size ”1460”. We refer to these signatures as ineffective because they do not provide unique information. Due to this reason, it is essential to filter out ineffective association rules. A simple approach is to build a filter list to collect all known ineffective signatures.

F. Construction Module

As many association rules are extracted from the transaction databases by the ARM module, we have to carefully select association rules to build the most effective signatures. In addition, we have to find a new way (different from the traditional methods) to apply the signatures to real-time traffic classification, especially for the new packet sequence signatures. To address this issue, we developed a construction module.

1) Signature Construction: We create a set of distinct transaction databases for various applications, and each corresponds to many packets that have the same sequence number in flows. We then build the payload-based signatures for both traditional individual packet signatures and the proposed packet sequence signatures. Similarly, when we format the transaction database with flow-level data formatting, we can build the feature-based packet sequence signatures.

Because the actual signature construction process is to find out which extracted association rules are more effective than others, we define a metric M to evaluate the effectiveness of an association rule. As each rule can be represented by an itemset $I = \{I_1, I_2, ..., I_m\}$, the metric M is defined as follows:

$$M = \frac{support(I)}{\prod_{i=1}^{m} \sigma_i} \tag{4}$$

where *support(I)* is the percentage of transactions that contain all the items in itemset I in the transaction database, and σ_i is the probability that item I_i appears in random cases. A higher M indicates a larger difference compared with random cases. Once the divisive hierarchical clustering process terminates, the metric of each association rule is obtained immediately. The rules are then sorted based on their values in a descending order, and the signature is constructed using the first k rules. So far, we are able to represent the feature-based packet sequence signatures and the payload-based individual packet signatures.

For the payload-based packet sequence signatures, further processing is required for a series of individual packets. Suppose we have built signatures for n successive packets, respectively, and every signature contains k association rules. Specifically, the transaction dataset for the n successive packets are built from the same set of flows, in which every flow is marked with a flow number. Then we use the flow number as the transaction id *TID*. We refer to *i-sequence rule* as the rule associated with i packets. Candidate sequence signatures, i.e., i-sequence ($i = 1, 2, ...n$) rules are obtained through the combination of the k association rules of every packet. Thus the total number of candidate i-sequence ($i = 1, 2, ...n$) rules is $C_n^i k^i$, and each of them can be represented as $I_s = \{(I_{11}, I_{12}, ...I_{1e}), (I_{21}, I_{22}, ...I_{2f}), ..., (I_{i1}, I_{i2}, ...I_{ig})\}$. Similarly, we define a metric M_s to evaluate the effectiveness of these candidate sequence rules.

$$M_s = \frac{support(I_s)}{\prod_{j=1}^{s} \sigma_j} \tag{5}$$

where *support(I_s)* is the percentage of transactions that contain all the items in itemset I_s in the transaction database, s is the total number of items in I_s, and σ_j is the probability item I_j appears in the random cases. The sequence rule with a higher M_s is obviously more effective. Therefore, the candidate i-sequence ($i = 1, 2, ...n$) rules are sorted based on the value of M_s. Finally, the packet sequence signature is built using the first k' rules.

2) Regex Engine Loading: For traditional signatures, the signature may be a string pattern or a regular expression matching individual packet payloads. Thus we only need to extract the payload content from the processed packet and then use multi-pattern matching algorithms or regular expression matching algorithms to identify the application. But if the packet sequence signatures are constructed for matching multiple sequential packets, we will have to record much more flow information, and the traditional matching methods do not work as before. In order to address this issue, we load a regex engine (DFA) with packet sequence signatures as regular expressions and design a new structure for flow tables. The new structure only records a DFA state number for each flow in the flow tables. Then at the arrival of a new packet, the DFA state number of the corresponding flow is updated. With this method, we can match a flow with packet sequence signatures with low memory consumption at a high speed.

In summary, our system design has the following advantages: (1) Both traditional signatures and our proposed packet sequence signatures can be constructed automatically with limited overhead. (2) The regex engine loading strategy used for packet sequence signatures matching with traffic flows is designed for practical use with low memory and low computation overheads.

V. Evaluation

In this section, we first compare the traditional single-packet signatures generated by the proposed system with other existing methods; we then present and evaluate the automatically generated packet sequence signatures. Our experiments are carried out on two datasets. Each is a combination of separated traces captured at several different sites with different network environments in two cities (Beijing and Harbin, China). Each trace is labeled with various types of pure application traffic, such that we can later evaluate the accuracy and recall of different methods. One dataset is used for automatically constructing signatures, the other is used for evaluating the generated signatures. The reason that we did not use public traffic traces is that they usually do not contain any application layer data, due to privacy concerns.

First, we compared the traditional signatures constructed by APSC with those obtained by L7-filter [13], LASER [5], AutoSig [6], and AppID [7]. As shown in TABLE.I, our system obtains more efficient signatures. The single packet signature of Bittorrent contains some long string patterns, and is easily identified by every method. For eDonkey, our signature is much shorter than that obtained by L7-filter or AppID. For FTP and POP3, although they have a similar matching pattern in traffic characteristics, we can use "+OK" and "TYPE.2331" to distinguish them. We have further verified the signatures with RFC documents [29]. QQ is the most popular communication tool in China, and once had more than 170 million users online simultaneously [30]. Our signature is more specific than any others. For DNS, we have one signature as the same as one obtained by AppID. We did not extract other ones identified by AppID in our trace. For SSL, the differences between our signature and L7-filter's are due to the deployment of new version of SSL. Our signature is constructed based on the now widely use of SSL3.0 [31].

TABLE II: Payload-based Signature Analysis for Applications.

APP	Protocol	Payload-based packet sequence signature
SMTP	TCP	.*->“^MAIL FROM” ->“^RCPT TO” ->“^DATA” ->“^FROM” ->“^TO” ->“^Subject”(c)
FTP	TCP	“^USER” ->“^PASS” ->“^ACCT” ->“^CWD” ->.*-> “^QUIT”(c)

Furthermore, to validate the effectiveness of the signatures automatically constructed by our system, we have evaluated them with the combination of three kinds of signatures: traditional individual packet signature (denoted as TIP), feature-based packet sequence signature (denoted as FPS), and payload-based packet sequence signature (denoted as PPS). As shown in TABLE.III, for the FPS signatures, the value of one byte in the signature represents for the packet size; while for the PPS signature of Skype, all the bytes in the signature are represented for the third byte of payloads in a continuous sequence of packets that belong to a Skype UDP flow. As shown in the table, our signatures achieve higher precision and recall in classifying these traffic.

Among them, Bittorrent, eDonkey, QQ, and SSL are classified with TIP signatures. The reason that we used TIP signatures is because each of them has strong individual packet signatures that can be well identified. FTP, POP3 and DNS are classified with the FPS signatures. As FTP and POP3 have the common matching pattern “’USER.*\x0D\x0A”, so

TABLE I: Traditional Signature Analysis for Applications.

APP	Protocol	Traditional signatures generated by APSC	Traditional signatures generated by other methods	
Bittorrent	TCP	‘‘^\x13BitTorrent protocol\x00{5}\x10\x00\x05’’(c&s)	‘‘^\x13BitTorrent protocol(ex|\x00\x00\x00\x00\x00)’’	AutoSig
		“^Get announce php?passkey==.*info_hash==”(c)	‘‘\x00\x00\x00\x00\x00”	
			‘‘/announce?info_hash\?.*&peer_id…’’	AppID
			‘‘^\x13BitTorrent protocol(ex|\x00\x00\x00\x00\x00) ’’	
eDonkey	TCP	“^\xE3|\xC5|\xD4.\x00{3}”(c&s)	‘‘^[\xc5\xd4\xe3-\xe5].?.?.?.?([\x01\x02 …’’	L7-filter
			‘‘^\xE3.\x00\x00\x00\x01\x10.*\x00[CHN]([|yourname)’’	AppID
			‘‘^(\xE39\xC5).\x00\x00\x00’’	
FTP	TCP	‘‘^(SIZE)|(TYPE.{2}331)|(PASS\x0D\x0A)|(USER.*\x0D\x0A)’’(c)	‘‘230 logged’’	LASER
			‘‘^USER .*\x0D\x0APASS”	AppID
			·‘^220(-9).*\x0D\x0A331’’	
POP3	TCP	‘‘^+OK’’(s)	‘‘USER.*PASS’’	AutoSig
		‘‘^(QUIT|PASS|USER|(\x43\x41\x50\x41)).*\x0D\x0A’’(c)	‘‘\+OK’’ (6 signatures in total)	
			‘‘^(USER|user) .*\x0D\x0APASS’’	AppID
			‘‘^\+OK.*\+OK’	
QQ	UDP	‘‘^(\x02|\x05.+\x03$)\|(\x04.{30}\x05.{9}\x00{3}.+\x03$)|(^\x05.{9}\x00{3})’’(c&s)	‘‘^.?.?\x02.+\x03$’’	L7-filter
			‘‘\x03$’’	AppID
DNS	UDP	‘‘^.{2}\x01\x00\x00\x01\x00\x00\x00\x00\x00\x00’’(c)	‘‘^.?.?.?.?[\x01\x02].?.?.?.?.?.?[\x01-?]’’	L7-filter
			‘‘^.{2}\x01\x00\x00\x01\x00\x00\x00\x00\x00\x00’’	AppID
			‘‘^.{2}(\x81|\x85)(\x82\x83\x80)\x00\x01\x00’’	
SSL	TCP	‘‘^[\x14\x15\x16\x17][\x01\x02\x03]’’(c&s)	‘‘^(.?.?\x16\x03.*\x16\x03|.?.?\x01\x03\x01?.*\x0b)’’	L7-filter

TABLE III: Performance Evaluation for Applications.

Method	APP	Protocol	Test set	Precision	Recall	Matching Signatures
TIP	**Bittorrent**	TCP	2000	100	100	‘‘^\x13BitTorrent protocol\x00{5}\x10\x00\x05’’(c&s)
TIP	**eDonkey**	TCP	2000	100	100	“^\xE3|\xC5|\xD4.\x00{3}”(c&s)
TIP	**QQ**	UDP	2000	100	97	‘‘^(\x02|\x05.+\x03$)\|(\x04.{30}\x05.{9}\x00{3}.+\x03$)|(^\x05.{9}\x00{3})’’(c&s)
TIP	**SSL**	TCP	2000	100	100	‘‘^[\x14\x15\x16\x17][\x01\x02\x03]’’(c&s)
FPS	**FTP**	TCP	2000	100	99	^\x010\x00E.\x008(c) ^\x01B\x048\x01F\x01D\x014(s)
FPS	**DNS**	UDP	2000	99	100	^[\x16-\x30]$(c) ^[\x28-\xF2]$(s)
FPS	**POP3**	TCP	2000	99	99	^\x007|\x006|\x010|\x000.{2}\x006\x006|\x008|\x000(c) ^\x057|\x05C|\x038|\x01D\x000(s)
PPS	**Skype**	UDP	2000	100	100	^(\x02+[\x0d\x1d\x2d\x3d\x4d\x5d\x6d\x7d]*)|(\x02+[\x0f\x1f\x2f\x3f\x4f\x5f\x6f\x7f]+[\x0d\x1d\x2d\x3d\x4d\x5d\x6d\x7d])|(\x02+[\x05\x1d\x25\x35\x45\x55\x65\x75]+[\x0d\x1d\x2d\x3d\x4d\x5d\x6d\x7d])(c&s)

they are difficult to be distinguished by traditional individual packet signatures. Note that AppID [7] has presented a less-desirable classification result for FTP and POP3 traffic with traditional methods. Due to this reason, we use our feature-based packet sequence signatures to classify them. Our experimental results show that the FPS signatures are very effective. For DNS, only traditional signatures for one direction (initiator) can be generated by APSC. If we use traditional signatures for classification, there will be a great loss in recall. From TABLE.III, we can see that, although the feature-based signatures for DNS are not sufficiently robust, the precision and recall are still high. This is because a DNS flow usually has only one packet. Skype is arguably the most popular VoIP application, and many methods have been developed to identify its traffic [32]–[38]. Due to the various kinds of communication modes in Skype, such as voice-calls, skypeOut, file transfer, and video conferencing, it is difficult to accurately identify them from mixed traffic. In [5], LASER was used to generate signatures for Skype traffic but failed with no clear signature found. However, the proposed system is able to discover the effective PPS signature for all types of Skype media traffic. Furthermore, we present PPS signatures for another two applications as well in TABLE.II. Due to the limited space, more experimental evaluation for the APSC system will be presented in our future work.

Finally, we evaluate the proposed system to validate whether our design goals discussed in Sec.IV have been achieved. (i) Our experimental results show that the proposed system obtains a high classification accuracy. (ii) Our unique design and implementation help us reduce system overhead and memory cost. The regex-engine-based signature matching method also improves the classification speed. (iii) Our packet sequence signatures enable us to achieve early identification, such that we can exploit the interactive communication in the negotiation phase of an application, before its actual data transmission. (iv) Our packet sequence signatures also have relative long-term effectiveness, because the packet sequence representing the interactive communications are rarely changed, even though some string patterns in a packet may be changed frequently. (v) As our system can construct signatures for traffic each direction of traffic flows, so the constructed signatures are able to achieve unidirectional identification.

VI. Conclusion

In this paper, we have first presented several crucial observations based on our experiences, which motivated us to further investigate payload-based and feature-based *packet sequence signatures*. We believe this first work applies this idea successfully to automatically build application signatures. The main advantages of the proposed scheme are: it is able to identify more challenging applications compared with existing solutions; it can also deal with frequent updates of applications with automatic signature generation with limited overhead. We have designed and implemented the proposed system, which can build both traditional single-payload signatures and the proposed new packet sequence signatures automatically. Our evaluation on the prototype system has shown that the proposed method is practical and effective.

Acknowledgment

This work was supported by the National Key Technology R&D Program of China under Grant No.2012BAH46B04. We would like to thank the reviewers for their insightful comments, and people of the organizations that support our work in evaluations.

References

[1] T. T. Nguyen and G. Armitage, “A survey of techniques for internet traffic classification using machine learning,” *Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.

[2] A. Callado, C. Kamienski, G. Szabó, B. Gero, J. Kelner, S. Fernandes, and D. Sadok, “A survey on internet traffic identification,” *Communications Surveys & Tutorials*, vol. 11, no. 3, pp. 37–52, 2009.

[3] N. Williams, S. Zander, and G. Armitage, “A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification,” *Proc. of ACM SIGCOMM*, 2006.

[4] P. Haffner, S. Sen, O. Spatscheck, and D. Wang, “ACAS: automated construction of application signatures,” in *Proc. of ACM SIGCOMM Workshop*, 2005.

[5] B.-C. Park, Y. J. Won, M.-S. Kim, and J. W. Hong, “Towards automated application signature generation for traffic identification,” in *Proc. of IEEE NOMS*, 2008.

[6] M. Ye, K. Xu, J. Wu, and H. Po, “Autosig-automatically generating signatures for applications,” in *Proc. of IEEE CIT*, 2009.

[7] Y. Wang, Y. Xiang, W. Zhou, and S. Yu, “Generating regular expression signatures for network traffic classification in trusted network management,” *Journal of Network and Computer Applications*, vol. 35, no. 3, pp. 992–1000, 2012.

[8] *Internet Assigned Numbers Authority (IANA)*. [Online]. Available: http://www.iana.org/

[9] Z. Li, G. Xia, H. Gao, Y. Tang, Y. Chen, B. Liu, J. Jiang, and Y. Lv, “Netshield: Massive semantics-based vulnerability signature matching for high-speed networks,” in *Proc. of ACM SIGCOMM*, 2010.

[10] R. Smith, C. Estan, S. Jha, and S. Kong, “Deflating the big bang: fast and scalable deep packet inspection with extended finite automata,” in *Proc. of ACM SIGCOMM*, 2008.

[11] R. Sommer and V. Paxson, “Enhancing byte-level network intrusion detection signatures with context,” in *Proc. of ACM CCS*, 2003.

[12] R. Smith, C. Estan, and S. Jha, “Xfa: Faster signature matching with extended automata,” in *Proc. of IEEE Symposium on Security and Privacy*, 2008.

[13] *L7-filter*. [Online]. Available: http://l7-filter.sourceforge.net/protocols/

[14] *Snort*. [Online]. Available: http://www.snort.org/

[15] A. McGregor, M. Hall, P. Lorier, and J. Brunskill, “Flow clustering using machine learning techniques,” *Passive and Active Network Measurement*, pp. 205–214, 2004.

[16] A. W. Moore and D. Zuev, “Internet traffic classification using bayesian analysis techniques,” in *Proc. of ACM SIGMETRICS*, 2005.

[17] T. Auld, A. W. Moore, and S. F. Gull, “Bayesian neural networks for internet traffic classification,” *IEEE Transactions on Neural Networks*, vol. 18, no. 1, pp. 223–239, 2007.

[18] B. Yang, G. Hou, L. Ruan, Y. Xue, and J. Li, “Smiler: towards practical online traffic classification,” in *Proc. of IEEE ANCS*, 2011.

[19] A. Este, F. Gringoli, and L. Salgarelli, “Support vector machines for tcp traffic classification,” *Computer Networks*, vol. 53, no. 14, pp. 2476–2490, 2009.

[20] N. Williams, S. Zander, and G. Armitage, “A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification,” *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 5, pp. 5–16, 2006.

[21] Y.-s. Lim, H.-c. Kim, J. Jeong, C.-k. Kim, T. T. Kwon, and Y. Choi, “Internet traffic classification demystified: on the sources of the discriminative power,” in *Proc. of ACM CoNEXT*, 2010.

[22] G.-L. Sun, Y. Xue, Y. Dong, D. Wang, and C. Li, “An novel hybrid method for effectively classifying encrypted traffic,” in *Proc. of IEEE GLOBECOM*, 2010.

[23] Y. Xue, D. Wang, and L. Zhang, “Traffic classification: Issues and challenges,” in *Proc. of IEEE ICNC*, 2013.

[24] *Weka*. [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/

[25] *Statistical Report on Internet Development in China*. [Online]. Available: http://www1.cnnic.cn/IDR/ReportDownloads/

[26] J. Han and M. Kamber, *Data mining: concepts and techniques*. Morgan Kaufmann, 2006.

[27] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, “From data mining to knowledge discovery in databases,” *AI magazine*, vol. 17, no. 3, p. 37, 1996.

[28] *Commview Tools*. [Online]. Available: http://www.tamos.com/products/

[29] *Request for Comments*. [Online]. Available: http://www.ietf.org/rfc.html

[30] *Tencent QQ*. [Online]. Available: http://im.qq.com/online/index.shtml

[31] *SSL 3.0*. [Online]. Available: http://tools.ietf.org/html/rfc6101

[32] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli, “Revealing skype traffic: when randomness plays with you,” in *Proc. of ACM SIGCOMM*, 2007.

[33] C.-C. Wu, K.-T. Chen, Y.-C. Chang, and C.-L. Lei, “Peer-to-peer application recognition based on signaling activity,” in *Proc. of IEEE ICC*, 2009.

[34] H.-S. Wu, N.-F. Huang, and G.-H. Lin, “Identifying the use of data/voice/video-based p2p traffic by dns-query behavior,” in *Proc. of IEEE ICC*, 2009.

[35] D. Bonfiglio, M. Mellia, M. Meo, N. Ritacca, and D. Rossi, “Tracking down skype traffic,” in *Proc. of IEEE INFOCOM*, 2008.

[36] K. Suh, D. R. Figueiredo, J. Kurose, and D. Towsley, “Characterizing and detecting relayed traffic: A case study using skype,” in *Proc. of IEEE INFOCOM*, 2006.

[37] M. Perényi, A. Gefferth, T. D. Dang, and S. Molnár, “Skype traffic identification,” in *Proc. of IEEE GLOBECOM*, 2007.

[38] J.-L. Costeux, F. Guyard, and A.-M. Bustos, “Qrp08-5: Detection and comparison of rtp and skype traffic and performance,” in *Proc. of IEEE GLOBECOM*, 2006.