Video Stream Identification for Traffic Engineering

Kazumasa Oida and Naoto Nakayama

***Abstract*—Video traffic represents a large fraction of Internet traffic. For efficient service provisioning and controlling traffic based on various policies, measuring detailed user behaviors, such as when, at what rate, how much, and between whom video data are transmitted, is critical. To collect such information, this paper deals with video stream identification, since source or destination address in a packet header may not be reliable for identifyingtrue source and destination. An unsupervised learning algorithm is proposed to perform stream identification. The algorithm overcomes two shortcomings of the existing clusterers. Experimental results show that the rate of correctly grouped classes achieved by the algorithm is 94%.**

***Index Terms*—Stream identification, video traffic, decay rate, unsupervised learning**

I. Introduction

Video traffic represents a large fraction of Internet traffic and will be increasingly prevalent on the Internet [1]. Measuring detailed user behaviors, such as when, at what rate, how much, and between whom video data are transmitted, is critical for efficient service provisioning, estimating capacity demand, and controlling traffic based on various policies. For example, this fine-grained information is used to detect the proliferation of video traffic within a social network, or to control traffic such that the net throughput is maximized, the inter-domain traffic is reduced, or the bandwidth is fairly shared among flows, users, or organizations.

IP addresses in packet headers are essential information to identify source and destination. Nowadays, however, there are cases where using IP addresses is not a reliable way for the following two reasons. First, the source and destination IP addresses in the packet header may be replaced by proxy servers [2], NAT devices [3], VPN technologies [4], etc. Second, the IP address of a host may be reassigned frequently. For example, every time remote PCs access private networks through remote-access VPNs, their IP addresses are reassigned [5]. Furthermore, since the Internet is composed of numbers of networks managed by various organizations, behaviors in a network, such as how IP addresses are replaced/reassigned, are basically invisible from other networks.

Let us consider stream identification, where two streams are identical if their source and destination hosts and streaming parameters are the same. Even if true source or destination IP address is unknown, an algorithm proposed in this paper shows that it is possible, with a high probability, to detect whether two streams are the same or not through traffic features. Then we can collect the information on when, at what rate, and how much video data are delivered between a certain source and destination pair. In other words, the stream identification is an alternative approach when source or destination address in a header is not reliable.

Currently, there are efficient traffic classification technologies that associate traffic flows with their applications [6], [7]. This paper assumes that the stream identification is performed after the application of a flow is classified as video streaming. Stream attributes for stream identification are different from those for application classification. In [8], end hosts are classified with a high true positive rate by using existing supervised learning algorithms, which uses the decay rates as stream attributes. This paper proposes an unsupervised learning algorithm, which uses the packet arrival rate, variance, and decay rates as stream attributes.

This paper is organized as follows: Section II summarizes the related work. Section III explains the method of measuring the decay rate. Section IV calculates the stream attribute set (the packet arrival rate, variance, and decay rates) and discusses the uniqueness and time invariance of the set. Section V proposes an unsupervised learning algorithm that overcomes two shortcomings of existing clusterers. Section VI evaluates performance of the algorithm from various points of view. Finally, Section VII presents the conclusions.

II. Related Work

Traffic classification associates traffic flows with applications that generated them [6],[7]. It is necessary to classify traffic for service differentiation, blocking malicious attacks, traffic engineering, etc. To infer Internet applications, deep packet inspection, which identifies byte strings associated with an application, are considered. Some approaches focus on extracting application signatures from payload content automatically [9]–[11]. This approach may not always be feasible since packets may be encrypted or encapsulated, privacy policy may prohibit payload inspection [12], or inspecting all packets flowing over high-bandwidth links may be computationally expensive.

Recently, many works have performed traffic classification based on flow parameters. This is because many Internet applications generate traffic with specific characteristics amenable to classification using machine-learning algorithms. In [13], the authors classify traffic flows by using the 5-tuple (source and destination IP addresses, source and destination ports and protocol) and flow duration and size. In [14], the authors use a Bayesian method to classify traffic flows by using measurements such as flow duration, flow bandwidth, and statistics on packet sizes and interarrival times. In [15], the authors classify flows by looking only at the lengths of the first packets in a flow.

Manuscript received August 3, 2012; revised September 5, 2012.

The authors are with the Department of Computer Science and Engineering, Fukuoka Institute of Technology, 3-30-1 Wajiro-Higashi, Higashi-ku, Fukuoka, 811-0295 Japan (e-mail: oida@fit.ac.jp).

DOI: 10.7763/IJFCC.2013.V2.167

Traffic classification for automated QoS management is discussed in [16].

III. Decay Rate

This section briefly introduces the decay rate [8] used for stream identification. Let $\{X_k\}$ be a time series, where X_k denotes the number of arriving packets during the k-th time interval of length δ. The m aggregated series of $\{X_k\}_{1\le k\le N}$, $\{X_k^{(m)}\}$, are obtained by dividing $\{X_k\}$ into blocks of length m and averaging the series over each block as

$$X_\ell^{(m)} = \frac{1}{m}\sum_{i=\ell m-m+1}^{\ell m} X_i\,,\ \ \ell = 1,2,\dots,\lfloor N/m\rfloor, \qquad (1)$$

where m is a positive integer, N is the size of series $\{X_k\}$, and $\lfloor x\rfloor$ is the largest integer that does not exceed x. The aggregated variance $V^{(m)}$ is the sample variance of $\{X_k^{(m)}\}$; i.e.,

$$V^{(m)} = \frac{1}{\lfloor N/m\rfloor - 1}\sum_{k=1}^{\lfloor N/m\rfloor}(X_k^{(m)} - \bar{X})^2, \qquad (2)$$

where $\bar{X} = \frac{1}{N}\sum_{i=1}^{N} X_i$. Hereafter, we assume that $m(\ge 1)$ is a real number. Let us consider the decay rate $\beta(m)$, the amount of change in $\log V^{(\lfloor m\rfloor)}$, defined by

$$\beta(m) = \log_{10} V^{(\lfloor 10^{\Delta} m\rfloor)} - \log_{10} V^{(\lfloor m\rfloor)}. \qquad (3)$$

In (3), Δ is a positive constant satisfying $\Delta = \log(m_{i+1}) - \log(m_i)$ for all i, where m_i is the i-th smallest level used for calculating decay rates. For simplicity, $\beta(m_i)$ is also described as β_i. In general, β_i fluctuates around $-\Delta$.

Decay rate $\beta(m)$ is influenced by various factors such as behaviors of users, protocols, computer software and hardware, communication devices, and resource (processors, bandwidth, etc.) competition among streams, etc. [17]. Computer hardware and communication devices in general have an impact on small timescale behavior of $\beta(m)$. Human users and protocols implemented by software affect larger timescale behavior of $\beta(m)$. Resource competition typically affects its small timescale behavior. However, heavy congestion may cause larger timecscale behavior in conjunction with the TCP congestion control mechanism. From the perspective of stream identification, resource competition is a factor that adds noise to stream attributes. This paper assumes that human users do not affect video streams. That is, they do not change/interrupt streams while collecting traffic data.

IV. Measurements

This paper uses arrival rate $\bar{X}$, variance $V^{(1)}$, and decay rates $\{\beta_i\}_{1\le i\le M}$ as stream attributes. Fig. 1 depicts ten samples of attribute set $\{\bar{X}, V^{(1)}, \beta_1, \dots, \beta_{20}\}$ for two streams. In the figure, $-300\times\bar{X}$ and $-60\times V^{(1)}$ are depicted at $\log(m) = -0.6$ and -0.3, respectively. For successful identification, each stream must have a unique and time-invariant attribute set. The time-invariance requires that ten samples should be almost the same. Fig. 1(a) shows ten attribute samples of a typical video stream. As shown in the figure, packet arrival rate $\bar{X}$ and variance $V^{(1)}$ are approximately constant. Furthermore, all decay rates β_i except for β_{19} and β_{20} are also roughly constant. Decay rates $\beta(m)$ at large m, such as β_{19} and β_{20}, are mostly unstable since from (2), $V^{(m)}$ is calculated with $\lfloor N/m\rfloor$ samples.

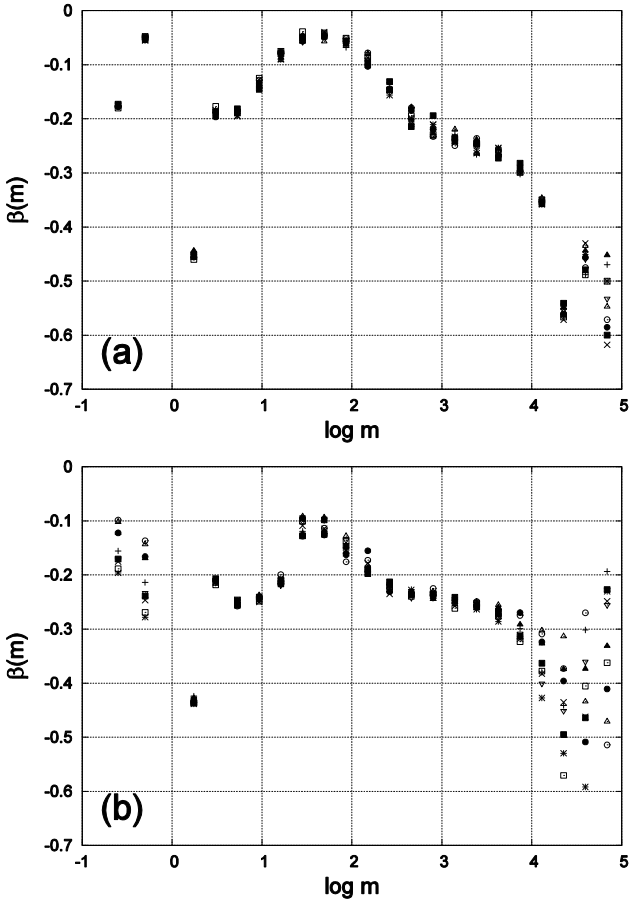

Fig. 1. Ten samples of attribute set$\{\bar{X}, \boldsymbol{V^{(1)}}, \boldsymbol{\beta_1}, \dots, \boldsymbol{\beta_{20}}\}$are depicted for two streams (a) and (b). $-300\times\bar{X}$and $-60\times\boldsymbol{V^{(1)}}$ are shown at $\mathbf{log}(m) = -0.6$ and-0.3, respectively. $\mathbf{log}(\boldsymbol{m_i}) = i\boldsymbol{\Delta}$ and $\boldsymbol{\Delta}\approx 0.24$.

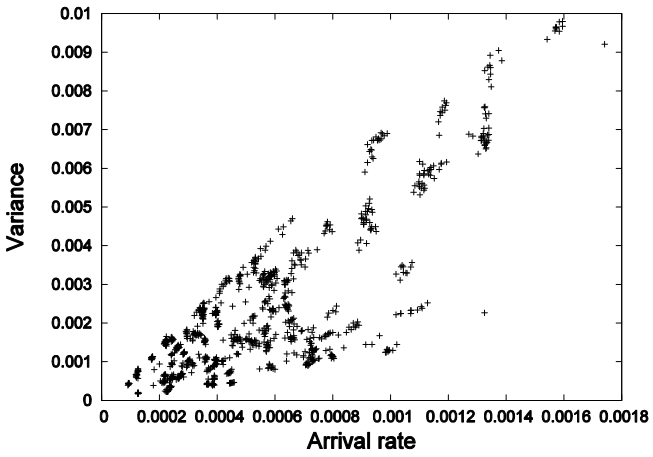

Fig. 2. Variance $\boldsymbol{V^{(1)}}$ is positively correlated with arrival rate $\bar{\boldsymbol{X}}$.

Meanwhile, Fig. 1(b) shows the case of variable packet arrival rate $\bar{X}$. By looking closely at all points of $\bar{X}$ and $V^{(1)}$ in the figure, it can be seen that $V^{(1)}$ increases with $\bar{X}$. In other words, $\bar{X}$ and $V^{(1)}$ are highly correlated. Fig. 2 explains

their relationship obtained with 1000 samples of $\bar{X}$ and $V^{(1)}$. The correlation coefficient is 0.81 in this case. Whereas, β_i is approximately uncorrelated with $\bar{X}$ or $V^{(1)}$. From Fig. 1(b), ten samples of β_i are not largely different if $i \leq 16$. Therefore, from the viewpoint of identification, either $\bar{X}$ or $V^{(1)}$ may be dispensable, but β_i provides effective information that is different from that provided by $\bar{X}$ and $V^{(1)}$.

Fig. 3 compares attribute sets of two streams flowing from the same TV site to different clients connected to the same switching hub. Traffic data are collected at the same point in the network. Let us focus on stable attributes $\bar{X}$, $V^{(1)}$, and $\{\beta_i\}_{1\leq i\leq 16}$. It can be seen that Figs. 3(a) and 3(b) exhibit very similar attribute sets (which implies that stream identification is not easy). There is a slight difference in β_1 to β_3 between (a) and (b). Compared with the difference between Figs. 1(a) and 1(b), the difference between Figs. 3(a) and 3(b) is far smaller for two reasons. First, a client affects decay rates only through the way it sends acknowledges (e.g., TCP ACKs). Second, two streams from a server to two clients in Fig. 3 flow along the same path since clients are connected to the same hub. Whereas, two streams in Fig. 1 do not take the same path since they are not located closely to each other.

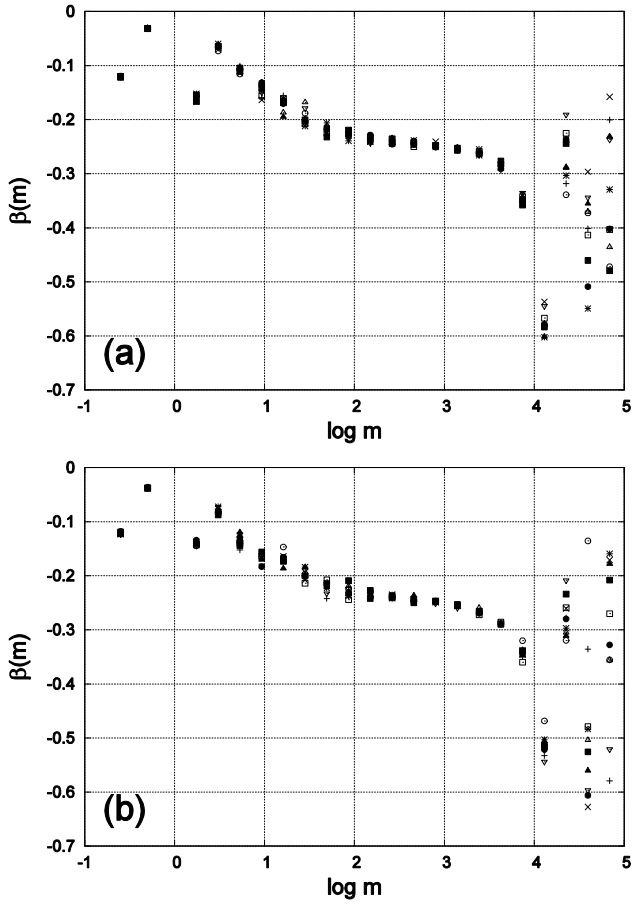

Fig. 3. The same TV channel is accessed by two clients: (a) Dell PC (Windows 7) and (b) Mac mini (MacOSX ver. 10.5).

V. Unsupervised Learning

A. Datasets

TV channels provided by content delivery networks, Web hosting companies, live streaming video platforms, etc. are accessed with Flash Player, Windows Media Player, or Silverlight. Any two streams do not originate from the same server. All video streams use the TCP protocol and most of them flow at constant rates. All packets are captured with WinDump [18]. Packets are collected at the same point in the network with the same personal computer (PC) (since decay rates especially at small aggregation levels are sensitive to changes in the collecting point and the collecting PC). Unless otherwise mentioned, attributes $\bar{X}, V^{(1)}, \beta_1, \ldots, \beta_M$ are computed with parameter values in Table I.

TABLE I: Default Parameter Values.

Symbol	Value
interval δ (s)	10^{-5}
size of $\{X_i\}N$	6×10^6
size of $\{\beta_i\}M$	5
interval Δ	$\frac{\log(N/50)}{M+1}$
level m_1	10^{Δ}
level m_M	$10^{M\Delta}$

B. Clusterers

This subsection performs unsupervised learning with three best performed clusterers in Weka [19]: farthest first (FF) [20], expectation maximization (EM) [21], and single-linkage clustering in the hierarchical clusterer (HC) [22]. The evaluation criterion is the rate of correctly grouped classes (CGR), where a correctly grouped class (CGC) is defined as follows. Class x is correctly grouped if and only if all samples that belong to class x are grouped into one class. A class corresponds to a video stream in this paper. In this subsection, there are two samples for each class.

Table II shows CGRs achieved by three clusterers. When executing three clusterers, they require the number of classes (the EM algorithm can work without the number, and if the number is not given, it groups all samples into two classes for all cases in Table II). From the table, the CGR tends to decrease with the number of classes, and all CGRs at 30 classes are less than 80%. Meanwhile, their computation times sharply rise with the number of classes. In the table, clusterer HC takes 2.8 hours to obtain the CGR at 35 classes.

TABLE II: The CGR (%) as a Function of the Number of Classes. Computation Times are Shown in Parentheses. Symbol ">" Indicates "More Than."

	10	20	30	35
FF	85	78.5	70.3 (2.0 h)	(> 25 h)
EM	88	85.0	76.3 (19.6 m)	(> 25 h)
HC	89	75.5	64.3 (12.8 m)	68.6 (2.8 h)

C. Proposed Algorithm

The unsupervised learning algorithm proposed in this paper overcomes two shortcomings of the existing clusterers: the necessity of specifying the number of classes and long computation times. Furthermore, the algorithm can handle a large amount of input data by dividing the input data into multiple parts and processing all parts one by one. Note that three clusterers in the previous subsection do not have this capability.

The proposed algorithm is described in Algorithm 1. The input dataset of the algorithm is different from that of the three clusterers. Instead of calculating one sample $\{\bar{X}, V^{(1)}, \beta_i\}$ from a time series $\{X_k\}$, $L(>1)$ samples are obtained by dividing $\{X_k\}_{1\leq k\leq N}$ into L blocks of length

$\lfloor N/L \rfloor$ and computing one sample per block. Let C_i be the class of the i-th time series $\{X_k\}_{1\le k\le N}$, and let A_i be the set of L samples derived from the i-th time series (therefore, $|A_i| = L$). The input dataset is $\{C_i, A_i\}$, where values of $\{C_i\}$ are assigned such that $C_i \neq C_j$ for all i and j. The algorithm judges that C_i is equal to C_j if there are many similar pairs s_1 and s_2, where $s_1 \in A_i$ and $s_2 \in A_j$. The algorithm performs clustering by repeatedly merging two sets A_i and A_j whose classes C_i and C_j are judged to be the same.

Algorithm 1: Merging sample sets of the same class.

1. **input:** T_h, $\{C_i, A_i\}_{1\le i\le n}$
2. $k \leftarrow n$
3. **while** $k \ge 2$ **do**
4. Derive $\{cm(i,j)\}_{1\le i,j\le n}$ from $\{C_i, A_i\}_{1\le i\le n}$
5. Select randomly a pair $(p,q) \in \arg\max_{\{(i,j)|i<j, A_i\neq\emptyset, A_j\neq\emptyset\}} \frac{cm(i,j)+cm(j,i)}{|A_i|+|A_j|}$
6. **if** $\frac{cm(p,q)+cm(q,p)}{|A_p|+|A_q|} \ge T_h$ **then**
7. $k \leftarrow k-1$
8. $A_p \leftarrow A_p \cup A_q$
9. $A_q \leftarrow \emptyset$
10. **else**
11. $k \leftarrow 1$
12. **end if**
13. **end while**
14. **return** $\{C_i, A_i\}_{1\le i\le n}$

In Algorithm 1, the agreement of two classes is measured as follows. A classifier performs the cross-validation test (described below) to obtain the confusion matrix $cm(i,j)$, which indicates how many times class C_i is classified as class C_j. Diagonal element $cm(i,i)$ is the number of correctly classified class C_i samples and $\sum_\ell cm(i,\ell) = |A_i|$ (see the cross-validation test description). If $i \neq j$, $cm(i,j)$ indicates the number of times class C_i is misclassified as class C_j. However, if C_i and C_j are originally the same, misclassification should occur numbers of times. Therefore, the algorithm determines that $C_i = C_j$ if $cm(i,j) + cm(j,i)$ exceeds a threshold. Normalized $cm(i,j) + cm(j,i)$ is compared with threshold T_h since it increases with $|A_i| + |A_j|$.

In Algorithm 1, a classifier is invoked to perform the following cross-validation test.

Cross-validation: The input dataset is randomly partitioned into ten subsets. A single subset of the ten subsets is retained as the validation data for testing the algorithm, and the remaining nine subsets are used as training data. The cross-validation process is repeated ten times, with each of the ten subsets used exactly once as the validation data. The ten results are averaged to produce a single estimation.

Note that all samples are used for both training and validation, and each sample is used for validation exactly once.

VI. Experiments

In the experiments, the number of classes is 100 (a client accesses 100 servers) and L (the size of sample set $|A_i|$) is 10, unless otherwise mentioned. We use two datasets $\{C_i, A_i\}_{1\le i\le 100}$ and $\{C_i, A_i\}_{101\le i\le 200}$ measured in two different periods of time, where A_i and A_{i+100} are sample sets of the same class. For performance evaluations, values of $\{C_i\}$ are assigned such that $C_i \neq C_{i+100}$ for all i, and $\{C_i, A_i\}_{1\le i\le 200}$ is used as input of the proposed algorithm. Unless stated otherwise, classifier naive Bayes [23] computes the confusion matrix. We use the term type I error to indicate an error that occurs when two sample sets belonging to different classes are judged to be the same class. In this case, there are two incorrectly grouped classes (IGCs). This section measures the correctly grouped class rate (CGR) and the incorrectly grouped class rate (IGR).

A. Sample Size

Fig. 4 demonstrates performance of the proposed algorithm for three sample sizes N (the sampling periods are $N\delta = 50, 100, 200$ s). In the figure, there is no type I errors at $T_h = 0.35$ and all CGRs exceed 60%. As T_h decreases, however, type I errors change CGCs into IGCs. One type I error creates two IGCs if they have not been IGCs and at the same time reduces the number of CGCs by two if they have been CGCs. Therefore, when the sample size and the threshold are both small ($N = 5\times10^6$ and $T_h = 0.1$ or 0.15 in the figure), many type I errors create many IGCs and reduce the CGR significantly. If the sample size is large, the type I errors seldom occur. When $N = 10^7$ or 2×10^7, the CGR is greater than or equal to 90% at 0.15, 0.2, or 0.25.

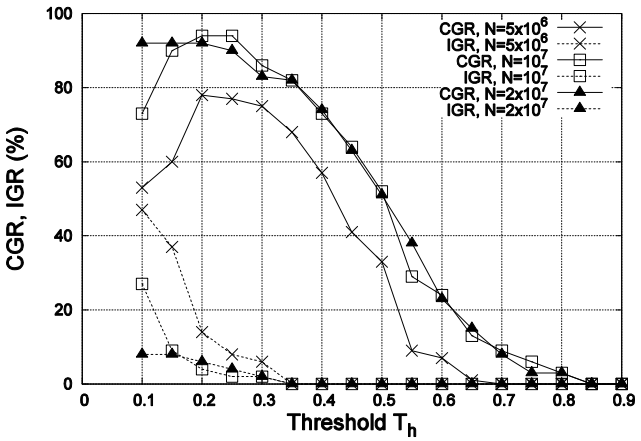

Fig. 4. The CGR and IGR for $N = 5\times10^6$, 10^7, or 2×10^7.

B. Sample Set Size

Let us consider the effect of L (the size of sample set A_i). Fig. 5 shows the results when N is fixed to $N = 5\times10^6$ and L changes. Since the naive Bayes algorithm does not work well when L is small, classifier k-nearest neighbor ($k = 1$) [24] is used at $L = 3$. From the figure, the largest CGR is almost the same for all L. By selecting $L = 5$, the CGR is approximately 80% over a wide range of T_h.

C. Number of Clients

If multiple clients are used, as shown in Fig. 3, each client leaves its signature in the stream attributes. A classifier uses this signature to improve classification correctness. Fig. 6 shows the CGR when the number of clients is 1, 2, or 3. The same 100 TV channels are used for the three cases. In the

case of two (resp. three) clients, the numbers of channels accessed by each client are 33 and 67 (resp. 31, 33, and 35). Three clients are on the same LAN, so that each sample $\{\bar{X}, V^{(1)}, \beta_i\}$ varies very slightly by changing the number of clients. As shown in Fig. 6(a), the CGR rises substantially by increasing the number from one to two. However, the increase rate at $T_h \in [0{,}15, 0.3]$ shrinks when the number further enlarges. Meanwhile, the CGR in Fig. 6(b) is already high over the range of [0,15,0.25] when the number of client is one. In this case, the CGR somewhat increases at someT_h values, but the largest CGR is unchanged.

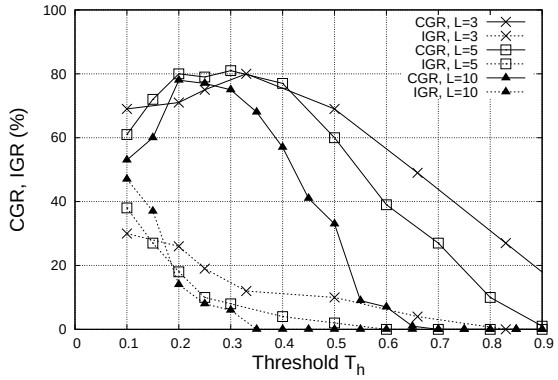

Fig. 5. The CGR and IGR for $\boldsymbol{L}$=3,5, or 10. $\boldsymbol{N = 5 \times 10^6}$.

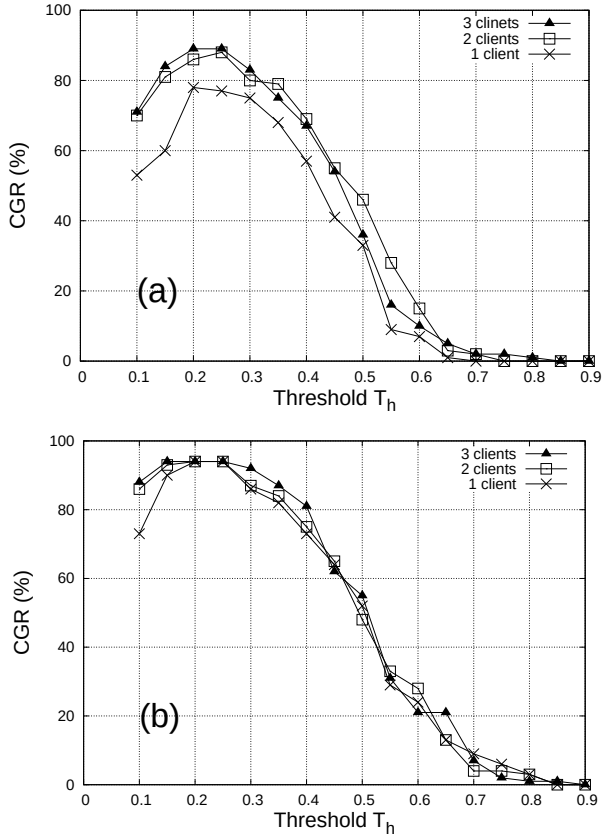

Fig. 6. The CGR when the number of clients varies. (a)$\boldsymbol{N = 5 \times 10^6}$ and (b) $\boldsymbol{N = 10^7}$.

D. Iterative Processing

When the input data are divided into multiple parts, the proposed algorithm can process all parts one by one. This is useful when the amount of input data is large or when new data are periodically given. In this experiment, instead of processing dataset $\{C_i, A_i\}_{1 \le i \le 200}$ all at once (batch processing), the dataset is divided in two and then processed such that the input for the first execution of the algorithm is $\{C_i, A_i\}_{i \in J}$, where $J \subset I \triangleq \{1{,}2, \ldots, 200\}$, and the input for the second execution is $\{\tilde{C}_i, \tilde{A}_i\} \cup \{C_i, A_i\}_{i \in I \setminus J}$, where $\{\tilde{C}_i, \tilde{A}_i\}$ isthe output of the first execution.

TABLE III: THE CGR AND IGR (CGR/IGR) FOR TWO PROCESSING SCHEMES AS FUNCTIONS OF T_h (OR T_h^2). $N = 10^7$.

	0.2	0.25	0.3	0.35
Batch	94/4	94/2	86/2	82/0
Iterative, J_1	91/10	90/2	87/2	80/2
Iterative, J_2	90/10	93/4	88/4	85/0
$J_1, T_h^1 = 0.4$	95/4	92/2	87/2	80/0
$J_2, T_h^1 = 0.4$	95/4	94/2	88/0	81/0

In Table III, set I is divided in two ways: $J = J_1$ and J_2, where $J_1 = \{1, \ldots, 50\} \cup \{100, \ldots, 150\}$ and $J_2 = \{1, \ldots, 150\}$. The table shows the CGR and IGR for two processing schemes. The table demonstrates that the batch processing slightly outperforms the iterative processing ("Iterative, J_1" and "Iterative, J_2" in the table). The iterative processing yields a larger IGR, whereby the CGR reduces. Let T_h^i be the threshold used in the i-th execution of the algorithm. Since the type I errors occur mostly in the first execution, T_h^1 should be greater than T_h^2. The table shows that the IGR decreases by using $T_h^1 = 0.4$.

E. Computation Time

The proposed algorithm is implemented in C and bash script. The bash script invokes a classifier in Weka to calculate a confusion matrix. The computation time of the proposed algorithm is governed by N_A (the number of non-empty sets A_i in the dataset) and N_m (the number of while-loop repetitions in Algorithm 1, or equivalently the number of times two sample sets are merged). We use N_A^i and N_m^i to indicate N_A and N_m at the i-th execution of the algorithm, respectively.

TABLE IV: COMPUTATION TIMES (S) OF THE PROPOSED ALGORITHM AT $T_h = 0.2$ (OR$T_h^2 = 0.2$) AND $N = 10^7$. NUMBER N_m (OR N_m^i) IS SHOWN IN PARENTHESES.

Conditions	Computation time	
	First	Second
Batch, $\{C_i, A_i\}_{1 \le i \le 100}$	5 (5)	
Batch, $\{C_i, A_i\}_{1 \le i \le 200}$	163 (100)	
$J_1, T_h^1 = 0.4$	31 (39)	92 (62)
$J_2, T_h^1 = 0.4$	48 (38)	94 (63)

Table IV shows computation times of the proposed algorithm. In the case of batch processing with input $\{C_i, A_i\}_{1 \le i \le 100}$, merging any two sample sets results in a type I error. The computation time is short in this case since $N_m (= 5)$ is small. Compared with the batch processing with $\{C_i, A_i\}_{1 \le i \le 200}$, the iterative processing with J_1 or J_2 reduces the computation time by 25% or 13%, respectively. The reduction is due to the numbers N_A, N_A^1, and N_A^2 since $N_m^1 + N_m^2 (= 101)$ for J_1 and J_2 is almost the same as $N_m (= 100)$for the batch processing. Whereas, $N_A = 200$ in the batch processing, $N_A^1 = 100$ and $N_A^2 = 161$ for J_1, and $N_A^1 = 150$ and $N_A^2 = 162$ for J_2. Note that the number of matrix elements the algorithm must compute is $(N_A)^2$ or

$(N_A^i)^2$.

VII. Conclusions

To identify pairs of source and destination hosts correctly, this paper proposed an unsupervised learning algorithm that performs stream identification by using the packet arrival rate, variance, and decay rates. The performance of the algorithm was evaluated by executing the algorithm under various conditions. The evaluation criterion was the rate of correctly grouped classes (CGR). The experimental results are summarized as follows:

1) The proposed algorithm overcame two shortcomings of the existing clusterers: the necessity of specifying the number of classes and long computation times. The computation time of the existing clusterers rose sharply with the number of classes, so that it was difficult to obtain the CGR at 35 classes or more. Whereas, the proposed algorithm calculated the CGR at 100 classes within three minutes when the sampling period $N\delta$ was 100 s. In this case, the CGR was 94%.
2) The CGR generally rose with an increase in the sample size N or in the number of clients, but the increase rate dropped quickly as N or the number of clients enlarges.
3) The algorithm could process input dataset all at once (batch processing) or one by one after dividing the dataset into multiple parts (iterative processing). The two processing schemes achieved roughly the same CGR, but the computation time of the iterative processing was shorter than that of the batch processing.

References

[1] *Cisco Visual Networking Index*, Forecast and Methodology, 2010–2015, June 2011.
[2] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. B. Lee, "Hypertext transfer protocol – HTTP/1.1," RFC 2616, June 1999.
[3] P. Srisuresh and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)," RFC 3022, 2001.
[4] J. C. Snader, *VPNs Illustrated: Tunnels*, VPNs, and IPsec, Addison-Wesley, 2005.
[5] S. Kelly and S. Ramamoorthi, "Requirements for IPsec Remote Access Scenarios," RFC 3457, January 2003.
[6] A. C. Callado, C. A. Kamienski, G. Szabo, B. P. Gero, J. Kelner, S. F. L. Fernandes, and D. F. H. Sadok, "A Survey on Internet Traffic Identification," *IEEE Communications Surveys and Tutorials*, pp. 37-52, 2009.
[7] A. Dainotti, A. Pescape, and K. C. Claffy, "Issues and future directions in traffic classification," *IEEE Network*, pp. 35-40, 2012.
[8] K. Oida and K. Yamashita, *Video Traffic Attributes for End Host Identification*, appeared in ICICT 2012.
[9] P. Haffner, S. Sen, O. Spatscheck, and D. Wang, "ACAS: automated construction of application signatures," in *Proc. of Mine Net*, pp. 197-202, 2005.
[10] H. Kim and B. Karp, "Autograph: Toward Automated, Distributed Worm Signature Detection," in *Proc. of USENIX Security Symposium*, pp. 271-286, 2004.
[11] Z. Li, M. Sanghi, Y. Chen, M. Kao, and B. Chavez, "Hamsa: Fast Signature Generation for Zero-day Polymorphic Worms with Provable Attack Resilience," in *Proc. of IEEE Symposium on Security and Privacy*, pp. 32-47, 2006.
[12] D. C. Sicker, P. Ohm, and D. Grunwald, "Legal issues surrounding monitoring during network research," in *Proc. of Internet Measurement Conference*, pp. 141-148, 2007.
[13] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINC: multilevel traffic classification in the dark," in *Proc. of SIGCOMM*, pp. 229-240, 2005.
[14] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," in *Proc. of SIGMETRICS*, pp. 50-60, 2005.
[15] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, "Traffic classification on the fly," *Computer Communication Review*, pp. 23-26, 2006.
[16] T. T. T. Nguyen, G. Armitage, P. Branch, and S. Zander, "Timely and Continuous Machine-Learning-Based Classification for Interactive IP Traffic," *IEEE/ACM Transactions on Networking*.
[17] K. Park and W. Willinger, *Self-Similar Network Traffic and Performance Evaluation*, Wiley-Interscience Published, 2000.
[18] Software WinDump. [Online]. Available: http://www.winpcap.org/windump/
[19] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *SIGKDD Explorations*, pp. 10-18, 2009.
[20] D. S. Hochbaum and D. B. Shmoys, "A best possible heuristic for the k-center problem," *Mathematics of Operations Research*, vol. 10, no.2, pp. 180-184, 1985.
[21] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society*, vol. 39, no. 1, pp.1-38, 1977.
[22] S. C. Johnson, "Hierarchical Clustering Schemes," *Psychometrical*, vol. 32, pp. 241-254, 1967.
[23] G. H. John and P. Langley, "Estimating Continuous Distributions in Bayesian Classifiers," in *Proc. of UAI*, pp.338-345, 1995.
[24] D. W. Aha, D. F. Kibler, and M. K. Albert, "Instance-Based Learning Algorithms," *Machine Learning*, pp. 37-66, 1991.