Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 6180527, 13 pages
http://dx.doi.org/10.1155/2016/6180527

Research Article

High-Performance Internet Traffic Classification Using a Markov Model and Kullback-Leibler Divergence

Jeankyung Kim,[1] Jinsoo Hwang,[1] and Kichang Kim[2]

[1] *Department of Statistics, Inha University, Incheon, Republic of Korea*
[2] *School of Information and Communication Engineering, Inha University, Incheon, Republic of Korea*

Correspondence should be addressed to Kichang Kim; kchang@inha.ac.kr

Received 6 August 2015; Revised 9 November 2015; Accepted 14 December 2015

Academic Editor: Francesco Palmieri

As internet traffic rapidly increases, fast and accurate network classification is becoming essential for high quality of service control and early detection of network traffic abnormalities. Machine learning techniques based on statistical features of packet flows have recently become popular for network classification partly because of the limitations of traditional port- and payload-based methods. In this paper, we propose a Markov model-based network classification with a Kullback-Leibler divergence criterion. Our study is mainly focused on hard-to-classify (or overlapping) traffic patterns of network applications, which current techniques have difficulty dealing with. The results of simulations conducted using our proposed method indicate that the overall accuracy reaches around 90% with a reasonable group size of $n = 100$.

1. Introduction

Traditional methods of network classification are either port-based or payload-based. However, both types of approaches are currently facing numerous challenges from the advanced techniques used to circumvent the firewalls of organizations and the increasing number of packet encryption techniques. Machine learning-based network classification techniques that use statistical information from network traffic data are currently emerging.

Machine learning techniques presented to analyze network traffic data so far can be categorized into the following three techniques. Supervised learning, a classification method, uses training traffic data with known application labels. Using this method, on the basis of training sample data, we can extract (or learn) patterns of applications and apply our knowledge to unlabeled test data. Unsupervised learning, also called clustering, on the other hand, is purely based on a predefined similarity measure of network traffic data. Clustering is one way of overcoming lack of enough traffic data for training. And finally so-called semisupervised learning [1] which combines both techniques exists.

Most learning approaches use packet duration, number of packets, average packet size, or interarrival time as statistical features. While several Markov models and hidden Markov models are used to extract information from packet sequence and directions, we utilize the Markov model proposed by Munz et al. [2], which uses a discrete time Markov model with a finite state space and gives relatively successful accuracy and complexity. In order to handle correlated data, which is a characteristic feature of network traffic data, Zhang et al. [3] proposed a "bag of flows" concept. Since the first few packets follow similar patterns per network applications or protocols, it would be better to form a certain-sized group and assign group membership in order to prevent wrong individual assignment due to a fluctuation.

In this paper, we generally follow a supervised learning techniques with a Markov model with states defined by the direction and size of the first four packets (in TCP connections, it is well known that the first four packets are most of time enough to capture the characteristics of the application [4]). Previous Markov modeling techniques, however, fail to differentiate among applications with overlapping, or hard-to-classify, traffic pattern as in IMAP and SMTP. Table 1 shows that both IMAP and SMTP include same state sequences: 0-4-1-4 and 1-4-1-4 (state 0 to state 3 are for client-to-server packets, and state 4 to state 7 are for server-to-client packets. Smaller state number at each case means smaller packet size.

Table 1: Overlapping state sequences of IMAP and SMTP.

IMAP		SMTP	
State sequence	Proportion	State sequence	Proportion
0-4-1-4	0.5564	0-4-1-4	0.4611
1-4-1-4	0.1134	1-4-1-4	0.0687
...	...	...	...

More details about this table and state numbering are in Sections 4 and 5.2), and these two sequences comprise more than 50% of packets in both applications (66.98% in IMAP and 52.98% in SMTP). Now consider a network flow with traffic pattern 0-4-1-4. With traditional Markov model classifiers, this flow will be classified as IMAP application because IMAP has the higher probability for this pattern. However, this flow also has a probability of more than 0.46 to belong to SMTP. The same argument can be applied to 1-4-1-4 pattern, and in fact we can say more than 52% of SMTP packets will be classified as IMAP packets under traditional Markov model classifiers.

Our solution to this issue is to build another Markov model for the testing flows and measure the similarity between the trained Markov model and the testing Markov model. In the training stage, we build and train a Markov model for each application. In the testing stage, we first collect a group of network flows, called a "bag of flows," that are believed to belong to the same application using only the port number, and then build a Markov model for each such group. After that, we use Kullback-Leibler divergence to measure the divergence between the Markov models of the training set and those of the test set. Finally, we classify a test group to an application whose Markov model has the smallest divergence from that of the test group. We verify the performance of our proposed method by means of theoretical and real data simulations.

The remainder of this paper is organized as follows. In Section 2, we review relevant related work on network classification. The theoretical background of our Markov model with Kullback-Leibler divergence and some evaluation measures for the classification used in our work are briefly explained in Section 3. Section 4 shows our proposed classification method with Kullback-Leibler divergence, while Section 5 presents the results of a theoretical simulation and real data-based analysis. Section 6 concludes and discusses further prospective developments.

2. Related Works

Traditional packet classification techniques are either port-based or payload-based approaches. Port-based approaches classify packets based on the ports used by the packets. However, nowadays, many applications, especially P2P applications, use unpredictable or dynamic ports, which limit the efficacy of this approach [5]. Payload-based, or Deep Packet Inspection (DPI), approaches look deep inside the packet to capture its application-specific pattern. However, this technique also suffers from challenges such as frequently changing packet formats, encryption of the packet payload, and load increments [6].

Machine learning approaches are being developed by researchers as an alternative to traditional approaches. A machine learning approach is either unsupervised or supervised. The unsupervised, or clustering, techniques start with unlabeled packets and classify them into different clusters. The simple K-means algorithm [4], in which each flow is represented by a point in a p-dimensional space, is one such technique. The first p packets of each flow are observed and the packet size of the ith packet becomes the coordinate at dimension i for this flow. The flows clustered close enough together are considered to belong to the same application. This technique has a problem such as the fact that it requires an application to dominate in at least one of the clusters, which is not always true, especially in close overlapping traffics such as SMTP and IMAP.

Supervised learning starts with known packet classes. Features such as packet size, packet direction, and packet inter-arrival time are extracted for each class. These feature vectors are used to build and train models that represent the classes. Various modeling techniques have been proposed. Roughan et al. [7] collected statistics on the feature vectors and classified unknown traffic via Nearest Neighbors (NN), Linear Discriminate Analysis (LDA), or Quadratic Discriminant Analysis (QDA) classification techniques. Moore and Zuev [8] applied naive Bayes techniques to map unknown traffic to preclassified traffic classes. These techniques are simple and effective but they generally consume considerable computing time or require large memory space to construct and maintain complex data structures. Also they tend to show very poor performance when the feature vectors are not clustered tightly enough. Crotti et al. [9] built *protocol fingerprints*, a PDF vector on packet size and packet interarrival time, for each traffic class, which can express the statistical characteristics of traffic classes in a compact and efficient way. Unknown traffic is then classified via *normalized threshold* techniques. However, they suggest to use a simple histogram of feature vectors for the *protocol fingerprints*, which is obviously too simplistic to capture the subtle difference between characteristically overlapping traffic classes.

Several means of improving supervised learning techniques have been attempted. Nguyen and Armitage [10] proposed taking packet samples from various locations during packet transmission to build multiple subflows. Most classification techniques capture either the entire flow or the first few packets of the flow as samples, but they claim that capturing the entire flow is time-consuming and detecting the beginning of packet transmission is not always possible. Instead, they capture packets in an intermittent way during packet transmission and use these multiple subflows to train their modeling system. An interesting idea of considering entire packet flow as a linear combination of a set of multiple component flows was introduced in [11–13]. They apply wavelet analysis to extract these components [11, 12] or apply ICA (Independent Component Analysis) [13] to identify the fundamental independent components and use them to classify target flow. Their target flow was abnormal flow generated by network attacking programs, but their techniques are general enough to be applied to any network flow. Reference [14] shows another approach of classifying abnormal packet flow. Their technique initially trains the classifier with normal traffic data only and then the classifier evolves in a dynamic way learning about anomalous behaviors using the Discriminative Restricted Boltzmann Machine. To overcome the limitation of a single technique, hybrid approach has been suggested in [15, 16]. Chen et al. [15] combine hardware classifier based on network processors with software classifier based on Flexible Neural Tree technique, while Ye and Cho [16] combine signature-based classifier with statistics-based classifier. By combining different classifying techniques we could expect faster or more accurate classifying performance.

Some other efforts have taken into consideration the time series characteristics of the transmitted packets in addition to their statistical properties. Palmieri and Fiore [17] plot recurring pattern of the training packets on RP (Recurrence Plot) and overlap the embedding vectors of the testing packets on top of RP measuring the distance of each pair of plotted points. The packets will be classified to the application from which the distance is the smallest. Dainotti et al. [18] and Mu and Wu [19] constructed Hidden Markov Models (HMMs) to represent the traffic classes while Munz et al. [2] built Markov Models (MMs) with eight states and four stages, which is much simpler than building HMMs but still effectively expresses the time series and statistical characteristics of the transmitted packets. Zhang et al. [3] proposed constructing a bag of flows and classifying it as a whole instead of classifying individual flow. Classifying a bag of flows as a whole has the advantage that the portion of the misclassified flows can be ignored if it occupies a smaller percentage of the bag, because the larger portion of the correctly classified flows determines the membership of the entire bag.

Above techniques improve the performance of supervised learning methods in various aspects. However they achieve that at the expense of other performance criteria. Capturing multiple subflows [10] and HMM [18, 19] both increase overall classification time considerably due to the extensive packet collection in the former case or due to the construction of the complex model in the latter. Each of the Markov Model by Munz et al. [2] and the Bag-of-Flow technique by [3] shows degenerated performance when the target traffic classes are overlapping in their traffic patterns. Hybrid approaches [15, 16] improve the performance but at the cost of increased classification time.

Our technique combines the Bag-of-Flow (BOF) technique in [3] with the Markov model approach in [2]. We believe that the Markov model is simple and powerful enough to grasp the characteristics of the traffic and that the BOF concept is essential for improving the classification accuracy. However, applying BOF technique directly to packet classification does not always produce the best results, especially when the target traffic classes show similar and overlapping characteristics, as in SMTP and IMAP. In such a case, direct application of BOF actually results in inferior performance to individual assignment (the details are given in Section 5.2). We propose the construction of another Markov model for the flows contained in the bag and to measure its similarity with the target Markov models. The flows in this bag are all assigned to the traffic class whose Markov model is most similar to the test Markov model.

3. Theoretical Background

3.1. Markov Models. A discrete-time Markov model $\{M(t), t = 1, \dots, m\}$ with finite state space $S = \{1, \dots, n\}$ is determined by the transition probability matrix $P = \{p_{ij}\}$ and initial probability distribution $\boldsymbol{\pi} = \{\pi_1, \dots, \pi_n\}$ of state space S, where

$$\begin{aligned} p_{ij} &= P\left(M(t+1) = j \mid M(t) = i\right), \quad i, j \in S, \\ \pi_k &= P\left(M(1) = k\right), \quad k \in S, \ \sum_{k=1}^{n} \pi_k = 1. \end{aligned} \tag{1}$$

We build separate Markov models for each known network traffic application in the training phase. Let $M_k^{\text{TR}}(t)$ denote a Markov model constructed from the training data of application k. Empirical estimation of transition probability and initial probability distribution can be done in a way similar to that of Munz et al. [2]. In the testing phase, we can proceed either by individual connection or by grouping correlated connections. For each group of connections, we build Markov models as was done in the training phase. Let $M_g^{\text{TE}}(t)$ denote a Markov model constructed from the testing data of group g. In the following section, we provide a dissimilarity measure between two Markov models.

3.2. Kullback-Leibler Information. Kullback-Leibler information is a well-known dissimilarity measure used between two probability distributions [20]. Let $\{x_1, \dots, x_n\}$ be a set of n observations drawn randomly from an unknown true probability distribution function $G(x)$, and let $F(x)$ be an arbitrary probability distribution function. We assume that the goodness of the model defined by $F(x)$ is assessed in terms of the closeness as a probability distribution to the true distribution $G(x)$. Akaike [21] proposed the use of the following Kullback-Leibler information (or divergence):

$$I(G; F) = E_G\left[\log\left\{\frac{G(X)}{F(X)}\right\}\right], \tag{2}$$

where E_G represents the expectation with respect to the probability distribution G. Kullback-Leibler information or relative entropy G with respect to F can be represented in discrete models as

$$I(g; f) = \sum_x g(x) \log\left\{\frac{g(x)}{f(x)}\right\}, \tag{3}$$

where g and f are probability mass functions of G and F, respectively.

3.3. Evaluation of Classification Methods. Classification performance can be calculated by measuring "error rates" or misclassification probabilities. In binary classification, it is well known that the total probability of misclassification (TPM) is given by

$$\begin{aligned}\mathrm{TPM} &= \Pr\left(\text{classified as } 2 \mid \pi_1\right)\Pr\left(\pi_1\right)\\ &\quad + \Pr\left(\text{classified as } 1 \mid \pi_2\right)\Pr\left(\pi_2\right),\end{aligned} \tag{4}$$

where $\Pr(\pi_k)$ represents the prior probability of class k.

In our work, we use recall and precision, which are used in [2], and F-measure to evaluate per-class performance, as in [3]. One has

(i) $\mathrm{recall}_k = \Pr(\text{classified as } k \mid \text{true class } k)$;

(ii) $\mathrm{precision}_k = \Pr(\text{true class } k \mid \text{classified as } k)$;

(iii) $F\text{-measure} = 2\times\mathrm{precision}\times\mathrm{recall}/(\mathrm{precision}+\mathrm{recall})$.

Let N be the total number of observations that belong to either one of two classes. Then, we can represent our evaluation measures in a simple 2×2 frequency table (see Table 2):

$$\begin{aligned}\mathrm{recall}_1 &= \frac{n_{11}}{n_{1\cdot}},\\ \mathrm{recall}_2 &= \frac{n_{22}}{n_{2\cdot}}\\ \mathrm{precision}_1 &= \frac{n_{11}}{n_{\cdot 1}},\\ \mathrm{precision}_2 &= \frac{n_{22}}{n_{\cdot 2}}.\end{aligned} \tag{5}$$

Hence, for example, recall_1 is the proportion correctly classified as application 1 out of all the true application 1 connections, and $\mathrm{precision}_2$ is the proportion correctly classified as application 2 out of all those classified as application 2 connections. F-measure is the harmonic mean of recall and precision, which hopefully represents the overall accuracy.

On the basis of the empirical measures, we can also estimate the TPM as follows:

$$\begin{aligned}\widehat{\mathrm{TPM}} &= \left(1-\mathrm{recall}_1\right)\frac{n_{1\cdot}}{N} + \left(1-\mathrm{recall}_2\right)\frac{n_{2\cdot}}{N}\\ &= \frac{n_{12}+n_{21}}{N}.\end{aligned} \tag{6}$$

4. Classification Using Kullback-Leibler Information

In this paper, we focus first on two applications, SMTP (port 25) and IMAP (port 143), whose traffic patterns are difficult to distinguish using existing classification algorithms, and later extend our technique to other applications in Sections 5.3 and 5.4. Bernaille et al. [4] showed that the first four packets of a TCP connection are sufficient to classify known applications with high accuracy; therefore, we build our Markov model using only the first four packets exchanged. State space is defined by a combination of direction of packets and four payload intervals: [0, 99], [100, 299], [300, MSS-1], and [MSS] (we follow the same payload intervals as in [2]. These intervals have been chosen because they emphasize well the difference in traffic feature vectors among the various applications [2]). The value of the Maximum Sequence Size (MSS) is often exchanged in a TCP connection. Since the direction is either from client-to-server or server-to-client, each stage can have $4 \times 2 = 8$ different states. Thus, our model becomes a four-stage left-right Markov model with state space $S = \{0, 1, \ldots, 7\}$. States 0–3 represent payload length intervals from client-to-server while states 4–7 represent those of server-to-client. For example, state sequence 0-4-1-4 means the following: client sends a 0–99 byte packet first (after the handshake), the server responds with a 0–99 byte packet, the client then sends a little larger 100–299 byte packet, and finally the server responds with a 0–99 byte packet.

Table 2

	Classified as class 1	Classified as class 2	Total
True class 1	n_{11}	n_{12}	n_1
True class 2	n_{21}	n_{22}	n_2
Total	$n_{\cdot 1}$	$n_{\cdot 2}$	N

By investigating the state sequences of SMTP and IMAP in training data, we find that 0-4-1-4 is the dominant pattern in both applications. Other common patterns, such as 1-4-1-4 and 0-4-0-4, also exist in both applications. In this section, we explain our classification model, which has only two common patterns: 0-4-1-4 and 1-4-1-4. We then extend our model in the simulation section to incorporate an extra unique pattern per application.

4.1. A Time-Variant Markov Model with Two Patterns. Suppose that we have two common patterns, 0-4-1-4 and 1-4-1-4, in both applications. Let M_k be the Markov model for application k (App k) and two patterns as observations, say O_1 (0-4-1-4) and O_2 (1-4-1-4), in our Markov models.

Assume that p_1 and p_2 are the proportions of O_1 in App 1 and App 2, respectively. Thus, the initial state probabilities are $\pi_0^k = p_k$, $\pi_1^k = 1 - p_k$, $\pi_l^k = 0$, $\forall l = 2, \ldots, 7$ for each model $k = 1, 2$. That is,

$$\boldsymbol{\pi}^k = \left(\pi_0^k, \ldots, \pi_7^k\right)' = \left(p_k, 1-p_k, 0, \ldots, 0\right)'. \tag{7}$$

Since we have four stages, we need three transition probability matrices in addition to the initial probability vector to compute a pattern probability. Let P_{ij}^k denote a one-step transition matrix from stage i to j in M_k; that is, $P_{ij}^k(s1, s2) = P(M_k(j) = s2 \mid M_k(i) = s1)$. On the basis of the frequency of observations O_1 and O_2 in each model, we can build three transition probability matrices, P_{12}^k, P_{23}^k, and P_{34}^k. The first and second transition probability matrices are shown below. P_{34}^k can be constructed in a similar fashion:

$$P_{12}^k = \begin{pmatrix} 0&0&0&0&1&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \end{pmatrix},$$
$$P_{23}^k = \begin{pmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \end{pmatrix}. \quad (8)$$

4.2. Theoretical Evaluation of Assignment. Let O_1 (0-4-1-4) and O_2 (1-4-1-4) be the only two observations of Markov models. Thus, for each connection, we can define $l_k(O_1) = P(M_k(1) = 0, M_k(2) = 4, M_k(3) = 1, M_k(4) = 4)$ and $l_k(O_2) = P(M_k(1) = 1, M_k(2) = 4, M_k(3) = 1, M_k(4) = 4)$. We can say $l_k(O_j)$ is the likelihood of O_j $(j = 1, 2)$ under model M_k. Each likelihood is computed as follows:

$$\begin{aligned} l_k(O_1) &= \pi_0^k P_{12}^k(0,4) P_{23}^k(4,1) P_{34}^k(1,4) = p_k, \\ l_k(O_2) &= \pi_1^k P_{12}^k(1,4) P_{23}^k(4,1) P_{34}^k(1,4) = 1 - p_k. \end{aligned} \quad (9)$$

4.2.1. Individual Assignment. Individual assignment is based on the likelihood of observations. Hence, the decision rule is

$$\text{Classify } O_j \text{ as } \begin{cases} \text{App 1} & \text{if } l_1(O_j) > l_2(O_j), \\ \text{App 2} & \text{if } l_1(O_j) < l_2(O_j), \\ \text{undecided} & \text{if } l_1(O_j) = l_2(O_j). \end{cases} \quad (10)$$

For simplicity, we assume $p_1 > p_2$ in the following performance calculations. Since $l_1(O_1) = p_1 > l_2(O_1) = p_2$ and $l_1(O_2) = 1 - p_1 < l_2(O_2) = 1 - p_2$, we assign observation O_1 to App 1 and O_2 to App 2. If $p_1 = p_2$, which is highly unlikely, we cannot determine its membership, so we leave it as undecided.

Table 3

	M_1	M_2
Recall	p_1	$1 - p_2$
Precision	$\dfrac{p_1}{p_1 + rp_2}$	$\dfrac{r(1-p_2)}{(1-p_1) + r(1-p_2)}$

Since the decision rule is given, we can determine the evaluation measure recall as follows:

$$\text{recall}_1 = \sum_{O_j \in M_1} P_1(O_j) I_{\{l_1(O_j) > l_2(O_j)\}} = P_1(O_1) = p_1, \quad (11)$$

$$\begin{aligned} \text{recall}_2 &= \sum_{O_j \in M_2} P_2(O_j) I_{\{l_2(O_j) > l_1(O_j)\}} = P_2(O_2) \\ &= 1 - p_2, \end{aligned} \quad (12)$$

where $P_k(O_j)$ is the proportion of O_j in model M_k and I_A is the set A indicator function. The above formula for computing recall can be used for more than two patterns. In order to determine precision, we need to estimate the ratio r of App 1 to App 2 and apply Bayes rule. We may assume $r = 1$ under no information regarding the abundance of both applications. A summary of evaluation measures is given in Table 3.

4.2.2. Group Assignment. We can form a correlated group of connections in several ways. Zhang et al. [3] used the concept of "flow," which consists of successive IP packets having the same five-tuple: [*src_ip, src_port, dst_ip, dst_port, protocol*]. They formed a BOF and assigned the BOF instead of individual connections.

In this paper, we use a simple concept of bag based on port number only. Constructing a group based on the same five-tuple is very time-consuming and may not be a good strategy in real-time classification problems. Therefore, our port-only based group assignment is fast and convenient even though it is slightly less correlated than five-tuple-based BOFs.

Let X_k denote the number of O_1 in a group with size n of model M_k. Then, X_k is a random variable whose distribution follows a binomial with parameter (n, p_k), that is, $X_k \sim \text{Bin}(n, p_k)$, and $P(X_1 > n/2)$ represents the probability of assigning a group to M_1. Thus, recall and precision can be computed similar to the individual assignment case, except that $P(X_1 = n/2)$ and $P(X_2 = n/2)$. We call $P(X_k = n/2)$ undecided$_k$, which means that we cannot assign such a group under model M_k (see Table 4).

We propose three group assignment methods: Majority, Kullback-Leibler, and 4096d. Majority assignment is based on the voting of each individual assignment, Kullback-Leibler assignment is our main proposed method, and 4096d assignment can be another reasonable candidate method based on Euclidean distance in 4096 dimensions. We explain the Kullback-Leibler and 4096d methods further in the next subsection.

Table 4

	M_1	M_2
Recall	$P\left(X_1 > \frac{n}{2}\right)$	$P\left(X_2 < \frac{n}{2}\right)$
Precision	$\frac{P(X_1 > n/2)}{P(X_1 > n/2) + r\{P(X_2 > n/2)\}}$	$\frac{rP(X_2 < n/2)}{P(X_1 < n/2) + rP(X_2 < n/2)}$
Undecided	$P\left(X_1 = \frac{n}{2}\right)$	$P\left(X_2 = \frac{n}{2}\right)$

4.2.3. Kullback-Leibler and 4096d Methods

Kullback-Leibler. For each bag of connections, we build a Markov model M^{TE} and measure divergence by computing Kullback-Leibler information $I(M^{\mathrm{TE}}; M_1)$ and $I(M^{\mathrm{TE}}; M_2)$. For $k = 1, 2$,

$$I\left(M^{\mathrm{TE}}; M_k\right) = \sum_{j=1}^{4096} l^{\mathrm{TE}}\left(O_j\right) \log \frac{l^{\mathrm{TE}}\left(O_j\right)}{l_j^{\mathrm{TR}}\left(O_j\right)}, \quad (13)$$

where $l^{\mathrm{TE}}(O_j)$ and $l_j^{\mathrm{TR}}(O_j)$ are the likelihoods of O_j under the model in test and training data, respectively. Since our Markov model consists of four stages and eight states, we have $8^4 = 4096$ possible observations in test data ($O_j = 1, \ldots, 4096$). To avoid division by zero, we use $l_k^{\mathrm{TR}}(O_j) = 10^{-5}$ for such j with $l_k^{\mathrm{TR}}(O_j) = 0$ and $l^{\mathrm{TE}}(O_j) \neq 0$.

If $I(M^{\mathrm{TE}}, M_1) < I(M^{\mathrm{TE}}, M_2)$, then we assign a group of testing connections to M_1. The evaluation measures recall and precision can be calculated in a similar fashion to the Majority case.

4096d. We can map each BOF of size n to a point in 4096-dimensional space, because we have $8^4 = 4096$ possible observations. For example, if a size-10 test BOF consists of $4O_1$, $2O_2$, $1O_4$, and $3O_5$, it can be represented as a point in 4096-dimensional space like $(4, 2, 0, 1, 3, 0, \ldots, 0)$. After suitable standardization, we use the Euclidean distance to determine the membership of a test BOF.

Theoretical considerations for handling more than two patterns can be easily made. Trinomial or multinomial distributions are needed instead of binomial distributions to compute evaluation measures. We include some of those simulation results in the next section.

5. Simulation Experiments

In this section, we evaluate the performance of our proposed method under various scenarios. First, several hypothetical model-based simulations are presented, followed by real network traffic-based results.

5.1. Model-Based Simulation. Suppose the proportions of observations for each application are given in Table 5.

In the following 4 hypothetical model-based simulations we choose values of p_1, p_2, q_1, and q_2 representing the real traffic sequences of ours and the more challenging scenarios, that is, $p_2 \approx q_2$. These parameters stand for the proportion of packet patterns for each application; that is, p_1 denotes the proportion of pattern 0-4-1-4 and q_1 denotes that of pattern 1-4-1-4 in application 1, while p_2 and q_2 stand for the proportions of the same patterns in application 2. For simplicity of notations, we use capital letter abbreviations to represent evaluation measures and subscript numbers to denote the application as usual:

Table 5

Application	Pattern			
	O_1 (0-4-1-4)	O_2 (1-4-1-4)	O_3 (0-4-1-0)	O_4 (0-4-0-4)
App 1	p_1	q_1	$1 - p_1 - q_1$	0
App 2	p_2	q_2	0	$1 - p_2 - q_2$

(i) R_k = recall$_k$, P_k = precision$_k$, F_k = F-measure$_k$, and U_k = undecided$_k$;

(ii) Maj = Majority, K-L = Kullback-Leibler.

Case 1 ($p_1 = 0.6$, $q_1 = 0.4$, $p_2 = 0.5$, and $q_2 = 0.5$ (see Table 6)). Case 1 represents a situation in which there are only two patterns. As expected, group assignment gives a better performance than individual assignment and both K-L and 4096d are better than Maj. An interesting observation in this case is the fact that the low values for R_2 in individual (50%) and Maj (46%) improve up to 86% in K-L as bag size becomes $n = 100$. Under M_2, which have 50% O_1 and 50% O_2, we wrongly assign up to half of true App 2 to App 1 in individual and Maj in group assignment. Nevertheless, even in that case, K-L performs well as group size increases.

The next three case studies deal with more than two patterns. Case 2 deals with a situation that is similar to Case 1 but with an extra unique pattern in application 2.

Case 2 ($p_1 = 0.6$, $q_1 = 0.4$, $p_2 = 0.5$, and $q_2 = 0.4$ (see Table 7)). The result shown in Table 7 shows that the individual assignment is even better than the group Maj method, but both methods give abysmal performances for R_2 and F_2. The K-L method gives the best performance among the competitors (99%).

Case 3 ($p_1 = 0.6$, $q_1 = 0.3$, $p_2 = 0.5$, and $q_2 = 0.4$ (see Table 8)). In Case 3, each application had its own unique pattern. K-L performs well in this case also but the performance gap between it and 4096d is smaller than that in Case 2. This tells us that K-L excels when a rare application specific pattern exists.

Case 4 ($p_1 = 0.56$, $q_1 = 0.11$, $p_2 = 0.46$, and $q_2 = 0.07$ (see Table 9)). Case 4 represents a situation that is close to real

Table 6

Group	n	Evaluation							
		R_1	P_1	F_1	U_1	R_2	P_2	F_2	U_2
Maj	10	0.6331	0.6268	0.6299	0.2007	0.3770	0.6940	0.4885	0.2461
	100	0.9729	0.6789	0.7997	0.0103	0.4602	0.9649	0.6232	0.0796
K-L	10	0.6331	0.6268	0.6299	0.0	0.6230	0.6294	0.6262	0.0
	100	0.8211	0.8582	0.8393	0.0	0.8644	0.8285	0.8461	0.0
4096d	10	0.6331	0.6268	0.6299	0.0	0.6230	0.6294	0.6262	0.0
	100	0.8689	0.8252	0.8465	0.0	0.8159	0.8616	0.8381	0.0
Individual		0.6	0.5454	0.5714	0.0	0.5	0.5555	0.5263	0.0

Table 7

Group	n	Evaluation							
		R_1	P_1	F_1	U_1	R_2	P_2	F_2	U_2
Maj	10	1	0.5004	0.6670	0.0	$10.4e-04$	1	$20.9e-04$	$1.4e-03$
	100	1	0.5	0.6667	0.0	$60.3e-25$	1	$10.2e-24$	$5.1e-24$
K-L	10	1	0.7415	0.8515	0.0	0.6513	1	0.7888	0.0
	100	1	0.9999	0.9999	0.0	0.9999	1	0.9999	0.0
4096d	10	0.6331	0.6513	0.6421	0.0	0.6611	0.6431	0.6519	0.0
	100	0.9729	0.9196	0.9455	0.0	0.9150	0.9712	0.8423	0.0
Individual		1	0.5263	0.6897	0.0	0.1	1	0.1818	0.0

Table 8

Group	n	Evaluation							
		R_1	P_1	F_1	U_1	R_2	P_2	F_2	U_2
Maj	10	0.8497	0.6927	0.7632	$10.0e-01$	0.3770	0.8884	0.5293	0.2461
	100	0.9999	0.6848	0.8129	$10.3e-05$	0.4602	0.9999	0.6303	0.0796
K-L	10	0.8463	0.8973	0.8710	0.0	0.9031	0.8546	0.8782	0.0
	100	0.9999	0.9999	0.9999	0.0	0.9999	0.9999	0.9999	0.0
4096d	10	0.7704	0.7866	0.7784	0.0	0.7910	0.7750	0.7830	0.0
	100	0.9857	0.9808	0.9832	0.0	0.9807	0.9856	0.9831	0.0
Individual		0.7	0.5833	0.6363	0.0	0.5	0.6250	0.5560	0.0

Table 9

Group	n	Evaluation							
		R_1	P_1	F_1	U_1	R_2	P_2	F_2	U_2
Maj	10	1	0.6884	0.8155	0.0	0.3057	1	0.4682	0.2417
	100	1	0.5909	0.7429	0.0	0.2413	1	0.3888	0.0665
K-L	10	1	0.9982	0.9991	0.0	0.9983	1	0.9991	0.0
	100	1	1	1	0.0	1	1	1	0.0
4096d	10	1	0.9830	0.9914	0.0	0.9827	1	0.9913	0.0
	100	1	1	1	0.0	1	1	1	0.0
Individual		1	0.6536	0.7905	0.0	0.47	1	0.6395	0.0

network traffic. The group assignment Maj performs worse than the individual in most of the measures, while K-L and 4096d perform as before. R_2 and P_1 of individual assignment are bad and worse in Maj but K-L and 4096d do exceptionally well in this case as well.

5.2. Real Data-Based Simulation. We retrieved traffic data from the packet traces in [22]. Our simulation system used the pcap library functions to extract valid TCP connections from the traces. We defined a valid TCP connection as being a packet exchange between a client and a server that starts with

Table 10: Composition of state sequence.

IMAP		SMTP	
State sequence	Proportion	State sequence	Proportion
0-4-1-4	0.5564	0-4-1-4	0.4611
0-4-1-0	0.2189	0-4-0-4	0.2870
1-4-1-4	0.1134	0-4-0-0	0.0906
0-4-0-0	0.0790	1-4-1-4	0.0687
0-4-1-1	0.0108	1-4-2-4	0.0150
0-4-0-4	0.0108	1-4-0-4	0.0141
$\cdots$	$\cdots$	$\cdots$	$\cdots$

proper three-way TCP handshakes and has at least four packets after them (currently our system eliminates flows with less than four packets. However, it is not difficult to extend our system to handle flows with less than four packets. We can simply add zero-length packets at the end of the flow when it does not contain all the four packets. Applications that produce less than four packets then can be characterized as flows ending with a number of zero-length packets). Among the packet traces, we singled out SMTP (port 25) and IMAP (port 143) connections. The trace files were huge and produced over 160,000 connections for SMTP and over 30,000 connections for IMAP. We have used tenfold cross-validation so that nine-tenths of the connections from each application selected via random selection is used to train the target Markov model and the remaining one-tenth is used to for testing purpose. For each connection, we have removed the three-way TCP handshake packets (SYN, SYN/ACK, and final ACK) and collected only the first four packets after the handshake. All acknowledgement packets are ignored.

Table 10 shows the state sequences (patterns) for each application in sorted order with the most frequent one at the top. Both applications had the 0-4-1-4 sequence as the most frequent sequence, with the other common sequences being 0-4-0-4, 0-4-0-0, and 1-4-1-4. Note that states 0 to 3 are for the client-to-server packets and states 4 to 7 are for the server-to-client packets with each state representing one of the four payload length intervals: $[0, 99]$, $[100, 299]$, $[300, \text{MSS-1}]$, and $[\text{MSS}]$ (Section 4). For example, 0-4-1-4 means the first packet was a client-to-server packet with size in $[0, 99]$, the second packet was a server-to-client packet with size in $[0, 99]$, the third packet was a client-to-server one with size in $[100, 299]$, and finally the fourth packet was a server-to-client one with size in $[0, 99]$. Let us summarize N_1 IMAP and N_2 SMTP application data collected in a given time interval as in Table 11.

Given a real network traffic observation O_j, we calculate the empirical likelihood under each model M_k,

$$\widehat{l}_k\left(O_j\right) = \widehat{\pi}^k(s1)\,\widehat{P}^k_{12}(s1, s2)\,\widehat{P}^k_{23}(s2, s3)\,\widehat{P}^k_{34}(s3, s4), \quad (14)$$

where $\widehat{\pi}^k(s1)$ is the proportion of state $s1$ in the first stage and $\widehat{P}^k_{ij}(si, sj)$ is an empirical transition matrix constructed from the total N_k connections. Evaluation measure, recall$_1$, can be computed as in (11) using empirical likelihood $\widehat{l}_k$ and

Table 11

	O_1	$\cdots$	O_j	$\cdots$	Total
IMAP	n_{11}/N_1	$\cdots$	n_{1j}/N_1	$\cdots$	N_1
SMTP	n_{21}/N_2	$\cdots$	n_{2j}/N_2	$\cdots$	N_2

$\widehat{P}_1(O_j) = n_{1j}/N_1$ instead. The other evaluation measures can be computed in a similar fashion.

Table 10 shows why SMTP and IMAP applications are difficult to classify using conventional network classification methods, such as the BOF technique in [3] and a plain Markov model approach in [2]. By eyeball computation, $l_{\text{SMTP}}(0414) < l_{\text{IMAP}}(0414)$ and $l_{\text{SMTP}}(1414) < l_{\text{IMAP}}(1414)$, so state sequences 0-4-1-4 and 1-4-1-4 of SMTP are misclassified as IMAP. Therefore, the recall rate of SMTP is at most 1–0.5298.

Group assignment using Maj does worse than the individual assignment in the recall rate of SMTP. Because more than half the percentage of SMTP state sequences are misclassified, majority counting will aggravate the situation. Table 12 shows the performance of individual and group assignment.

The Kullback-Leibler approach gives the best performance as in the model-based simulation. The recall rates of SMTP (R_2) with the Kullback-Leibler approach are well over 90% and close to 100% for a bag size of 100. The recall rates of IMAP (R_1) are also remarkably high. The values of R_1 for K-L (Kullback-Leibler) are somewhat lower than those for the Maj approach, but it should be understood that the high recall rate of IMAP in majority counting comes at a severe sacrifice of SMTP recall rate. The 4096d approach also shows strong recall rates, around 90%, for SMTP traffic. Its performance degrades, however, with IMAP traffic, showing around a 65% recall rate. It appears that simple Euclidean distance in 4096-dimensional space is not accurate enough to capture the characteristics of a traffic class whose traffic pattern heavily overlaps with another. Further, 4096d treats all dimensions equally in a sense, so it may not be sensitive enough to detect some unique rare patterns in certain applications.

Even though the Kullback-Leibler method performed well in the real data simulation, the computing time is still a concern in online network classification and decision-making. Table 13 presents a comparison of execution times for various algorithms. The execution times are normalized for 10,000 connections. The individual assignment and Maj show similar time requirements. The K-L approach is slightly faster than the 4096d method but is about roughly ten times slower than the individual assignment or Maj in the case of bag size of 100 connections, which seems to be a reasonable bag size (100 connections are a size that is large enough to present a representative state sequence distribution and small enough to declare the identity of some unknown traffic in due time). Most of the time is consumed in building the Markov model for the test data; computing the distance between two Markov models does not take significant execution time, contrary to our initial prediction. It turns out that the probability distribution of our total state-sequence space is very sparse,

Table 12: Evaluation results for real data.

Group	n	Evaluation							
		R_1	P_1	F_1	U_1	R_2	P_2	F_2	U_2
Maj	10	0.9148	0.6481	0.7587	0.0393	0.3598	0.8869	0.5119	0.1436
	100	1	0.6249	0.7692	0.0	0.3574	1	0.5266	0.0424
K-L	10	0.8853	0.9450	0.9142	0.0	0.9485	0.8921	0.9194	0.0
	100	0.9017	0.9933	0.9453	0.0	0.9939	0.9100	0.9501	0.0
4096d	10	0.7182	0.8688	0.7864	0.0	0.8916	0.7598	0.8204	0.0
	100	0.6553	0.8642		0.0	0.8970	0.7224		0.0
Individual		0.9001	0.6262	0.7386	0.0	0.4625	0.8223	0.5920	0.0003

Table 13

Bag size	Individual		Maj		K-L		4096d	
	time$_1$	time$_2$	time$_1$	time$_2$	time$_1$	time$_2$	time$_1$	time$_2$
10	0.0170	0.0131	0.0109	0.0098	0.4277	0.4194	0.8899	0.8913
50	0.0170	0.0131	0.0103	0.0098	0.1551	0.1311	0.2290	0.2326
100	0.0170	0.0131	0.0103	0.0098	0.1169	0.1016	0.1466	0.1475

Table 14: Recall rates for various protocols.

Group	n	Recall rates									
		R_{21}	R_{22}	R_{25}	R_{80}	R_{110}	R_{119}	R_{143}	R_{443}	R_{995}	R_{6881}
Maj	10	0.6083	0.9941	0.1375	0.9934	10.000	0.1260	0.9410	0.9088	10.000	0.9849
	100	0.7254	10.000	0.1453	0.9977	10.000	0.2100	10.000	0.9680	10.000	1.000
K-L	10	0.8259	0.9941	0.9370	0.9942	0.9845	0.4957	0.8689	0.9725	10.000	0.9953
	100	0.8883	10.000	0.9878	0.9984	10.000	0.7899	0.9344	0.9872	10.000	1.000
4096d	10	0.2678	0.9045	0.8492	0.9207	0.9988	0.0000	0.7247	0.9514	0.0000	0.9907
	100	0.3348	0.9074	0.9394	0.9508	10.000	0.0000	0.6395	0.9808	00.0000	1.000
Individual		0.5067	0.9912	0.1328	0.9857	0.9574	0.1617	0.9036	0.7964	0.9758	0.9258

meaning that most of them are zeroes. Therefore, we can significantly reduce the K-L computing time.

5.3. Extending to Multiple Applications. We have extended our proposed approach to handle network classification problems in the presence of more than two applications. From the network traffic data repository we have collected most of the application protocols having enough traffics to analyze. As a result we ended up with 10 network protocols, which are FTP, SSH, SMTP, HTTP, POP, NNTP, IMAP, HTTPS, SPOP, and BitTorrent.

We have conducted an experiment to check whether our proposed method works well in the presence of other protocols. Tables 14 and 15 show the recall and precision rates of various models, respectively, for the chosen protocols. R_x in Table 14 is the recall rate for a protocol with port number x, while P_x in Table 15 represents the precision rate for the same protocol. BitTorrent protocol uses random port numbers, but we have collected and tested packets destined to port 6881 since our pcap files contain BitTorrent traffic of older version in which 6881 is known to be one of the most frequent port numbers (we discuss the problem of detecting BitTorrent packets in the presence of random port numbers in Section 5.4). Thus, R_{6881} and P_{6881} stand for the recall and precision rate of these representative BitTorrent packets. The recall rates of SMTP and IMAP are slightly different from those in Table 12 since each packet now is matched against 10 different models, instead of two models. It is clear that our techniques, KL-10 (Kullback-Leibler with bag size 10) and KL-100 (Kullback-Leibler with bag size 100), consistently show much better performance than other techniques. Other techniques show very poor recall rate for ports 21, 25, and 119. However, KL-10 and especially KL-100 produce close to 90% recall rate for all ports except for NNTP (port 119), which must be very tough to classify correctly as can be seen in the table. Still, KL-100 matches the packets of NNTP much better than other techniques, almost reaching 80% recall rate with bag size 100 while others produce less than or around 20% recall rate.

Precision rates in Table 15 show the accuracy of the prediction of each model. Again the proposed KL model displays considerably high precision rates compared to others. To compute recall and precision rates, we need the distribution of predicted ports for each classification technique. For example, Table 16 shows the distribution of predicted ports for each protocol with KL-10 classification technique. The table shows the total number of connections belonging to each protocol in the first column. For example, 1792 connections were found to belong to port 21. The rest of the columns show the predicted port number for each protocol. For port 21, the KL-10 has predicted that 1480 connections belong to

Table 15: Precision rates for various protocols.

Group	n	Precision rates									
		P_{21}	P_{22}	P_{25}	P_{80}	P_{110}	P_{119}	P_{143}	P_{443}	P_{995}	P_{6881}
Maj	10	0.9536	0.9942	0.7287	0.9857	0.4095	10.000	0.5979	0.9873	0.9548	0.9328
	100	10.000	10.000	10.000	10.000	0.4876	10.000	0.5518	0.9984	0.9968	0.9213
K-L	10	0.9674	0.8753	0.8753	0.9990	0.6189	0.9404	0.9391	0.9928	0.9918	0.9432
	100	10.000	10.000	0.9378	10.000	0.7566	10.000	0.9872	0.9984	10.000	0.9874
4096d	10	0.7424	10.000	0.6578	0.9843	0.4635	0.0000	0.8667	0.4875	0.0000	0.6329
	100	0.8738	10.000	0.5793	10.000	0.4978	0.0000	0.9135	0.4952	0.0000	0.6909
Individual		0.7996	0.9644	0.5635	0.9467	0.3913	0.9985	0.6205	0.8952	0.9051	0.9024

Table 16: Predicted port distribution by KL-10.

Port	21	22	25	80	110	119	143	443	995	6881
21	1480				212	40				60
1792 (0.5580)	825.89				118.29	22.32				33.48
22		3396								20
3416 (0.2927)		994.01								5.85
25	30		15468		80		930			
16508 (0.0606)	1.82		937.36		4.85	56.36				
80	10			127416				430		300
128156 (0.0078)	0.08			993.84				3.35		2.34
110	100	60			24820	230				
25210 (0.0397)	3.97	2.38			985.35	9.13				
119	10				230	236				
476 (2.1008)	21.01				483.18	495.79				
143			400				2652			
3052 (0.3277)			131.06				869.06			
443				30				30396	260	570
31256 (0.0320)				0.96				972.67	8.32	18.24
995										290
290 (3.4483)										1000
6881	10							40		10732
10782 (0.0927)	0.93							3.71		995.36

port 21, 212 connections belong to port 110, 40 connections belong to port 119, and 60 connections belong to port 6881. Therefore, the recall rate is 1480/1792 = 0.8259 as shown in Table 14. The precision rate for port 21 can be computed by collecting all numbers in column 2 which shows the number of connections predicted to belong to port 21 from all the 10 protocols. 1480 connections out of 1792 connections destined to port 21 have been matched to port 21, 30 connections out of 16508 connections destined to port 25 have been matched to port 21, and so forth. However, since each protocol has different number of connections, we have to normalize the predicted connection number before adding them. We have normalized the connection number of each protocol to 1000 connections, and the table shows the normalized connection number below each connection number. The numbers within parentheses in the first column of Table 16 are normalization factors. Now the sum of normalized numbers in column 2 (which shows all the number of connections predicted to belong to port 21) is 861.698; therefore the precision rate of KL-10 technique for port 21 is 825.89/853.698 = 0.9674 as shown in Table 14.

5.4. Detecting BitTorrent Packets in the Presence of Random Port Numbers. Detecting P2P packets such as BitTorrent is a hard problem and has been investigated by numerous researchers. In this section, we describe how our technique can be applied to detect BitTorrent packets and provide some preliminary experimental results. Since our technique needs a set of flows believed to belong to the same protocol, in this case BitTorrent, we have collected packets coming out from the same host to build a bag of flow and applied our technique to it. We have identified hosts that exchange packets with more than 10 different peers within relatively short time period, in our case 1000 seconds. There were 1049 such hosts in the pcap files used in the experimentation. Since the pcap files do not contain payload portion, we cannot tell exactly which traffic is due to true BitTorrent application (another approach of collecting traffic for BitTorrent detection would

Table 17: Port distribution of the ows from the 1049 hosts.

Port	21	22	25	80	110	119	143	443	995	BT	Other
True	273	38	8278	0	136	51	14	755	1	943	26634
Pred.	744	145	8036	4626	294	191	189	11115	84	11699	0

be generating BitTorrent packets by ourselves and combine them with existing pcap files. But it is very hard to generate realistic BitTorrent traffic in this way, and the BitTorrent traffic generated this way would consist of very simple traffic patterns whose classification would become a trivial task). Instead we have assumed port 6881 through 6899 are BitTorrent ports and collected packets with these ports as BitTorrent traffic. Packets with these ports have very high chance of being BitTorrent packets (in older version of BitTorrent, whose traffic our pcap files contain, 6881–6899 are known to be the port range that BitTorrent hosts are using.), and the purpose of our experimentation is to see how much of them are classified as BitTorrent packets by our KL model.

The 1049 hosts were producing 37123 flows, or connections, and the port distribution of them are shown in Table 17. About 30% of them were destined to one of the well-known ports that our classification system can recognize, while the rest (70% of the whole traffic), categorized as "Other" in the table, were using random port numbers. The table also shows the prediction result by the proposed KL-10 method. Since each flow will always be matched to one of the 10 models, there is zero flow in "Other" category. Instead each category has been predicted to contain more than the actual number of flows belonged to it. We believe the prediction system will become more precise when it is equipped with more Markov Models for other missing protocols.

We are especially interested in the recall and precision performance for the case of BitTorrent traffic. Out of 1049 hosts, we have further identified 39 hosts that are producing traffic with ports in 6881–6899 range. The total number of BitTorrent flows in this category was 943. We were interested in how much of them are classified as BitTorrent by our prediction system and how precise our prediction is. The result is shown in Table 18 and summarized at the bottom of the table. "num of BT" in the table stands for "the number of BitTorrent flows," and "num of Non-BT" stands for "the number of non-BitTorrent flows." The table shows for each host the true number of BitTorrent flows and non-BitTorrent flows, respectively, and at the same time the predicted number of BitTorrent and non-BitTorrent flows. From the table we can see all of the BitTorrent flows are classified as BitTorrent in our detection system. Therefore the recall rate is 100%. However, the precision rate is 54.70%, a much lower figure than those in Table 15. The main reason is that our system has only 10 models, ports 21, 22, 25, 80, 110, 119, 143, 443, and 995 and BitTorrent, to classify the traffic. All other traffics that do not belong to one of these, those in the "Other" category in Table 17, still have to be classified into one of these models, lowering the precision rate. Particularly, a significant portion of them are classified as BitTorrent traffic as shown in Table 18. It might be that these flows are actually BitTorrent traffic, or they are another P2P traffic that has similar packet pattern as BitTorrent.

Table 18: Prediction result for BitTorrent hosts.

	True port		Predicted port	
	num of BT	num of Non-BT	num of BT	num of Non-BT
Host 1	25	110	125	10
Host 2	30	5	35	0
Host 3	75	8	83	0
Host 4	48	1	49	0
Host 5	14	2	16	0
Host 6	27	24	41	0
Host 7	62	5	67	0
Host 8	8	4	10	2
Host 9	13	1	1 4	0
Host 10	22	3	25	0
Host 11	3	33	10	23
Host 12	56	1	50	7
Host 13	6	19	10	15
...	...	...	...	...
Total	943	880	1724	99
Recall		1.0000		
Precision		0.5470		

The low precision rate could be a problem in real situation when we deploy our system in the gateway server. However, by increasing the number of models, we believe that the precision rate will improve. Also, since it captures unknown traffic as BitTorrent only by looking at the packet header, we can combine it with Deep Packet Inspection (DPI) technique to classify the traffic further. That is, instead of applying DPI to all the traffic, we can extract suspected BitTorrent traffic with our technique first and then apply DPI to these extracted ones.

There are other concerns when the current system is deployed in real situation. One is building and maintaining LRU (Least Recently Used) list to keep track of hosts for which to collect packets. Since we cannot keep track of all possible hosts, we use LRU to remove relatively inactive hosts from the monitoring list. How many hosts to keep in the LRU list, how many packets per host to collect, and so forth are questions to resolve in real world deployment. Too short LRU list will remove BitTorrent hosts prematurely from the list when a large number of non-BitTorrent hosts exchange packets before the BitTorrent host has the chance to send or receive the second packet. Too long LRU list obviously put too much overhead on the system. Another concern is the relatively slow classification time of Kullback-Liebler method.

Table 19

	O_1	$\cdots$	O_j	$\cdots$	O_c	Total
App 1	p_{11}	$\cdots$	p_{1j}	$\cdots$	p_{1c}	1
$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$
App$r$	p_{r1}	$\cdots$	p_{rj}	$\cdots$	p_{rc}	1

However, in a moderate speed network as in our test pcap file, the timestamps of captured packets show that there is enough time to analyze traffic pattern and classify them with Kullback-Liebler technique. Even in high speed network, classification task is a highly parallel process and with proper equipment such as parallel network processors our technique would still be a viable option.

6. Discussion and Conclusion

In this paper, we developed a novel classification method based on a Markov model with Kullback-Leibler divergence. Our primary goal was to develop a method that performs well on hard-to-classify network applications. Even though most of the previous methods of network classification perform well in most cases by using either correlated information of connections or a combination of a machine learning technique and Markov or hidden Markov models, they fail to produce convincing results when the patterns of connections of applications are similar.

We proposed a novel method that combines a flexible Markov model with Kullback-Leibler information and correlated traffic connection by grouping or bagging with the port number of applications. As our theoretical simulation and real data simulation show, our method outperformed the other methods in hard-to-classify situations, even though we did not cover all possible cases.

We recognize the slowness of the Kullback-Liebler approach (compared to *Individual* or *Majority* approach as shown in Table 13) as being one drawback of our technique. However, its high prediction success rate even among the overlapping traffic classes in terms of traffic patterns, as in SMTP and IMAP, is promising. Further, our technique is scalable in that its execution time increases linearly as the number of target classes increases, because once it builds the Markov model for the test data, measurement of its distance from each of the Markov models of the target classes can be done very quickly.

Our approach can be extended to more general multiclass problems. A $r \times c$ table can be set up to classify observations with c (> 2) patterns to one of the r (> 2) classes (see Table 19).

The basic idea of individual assignment is to compute $l_k(O_j), k = 1, \ldots, r,\ j = 1, \ldots, c$, and assign O_j to $\arg\max_{1\le k\le r} l_k(O_j)$ class for all $j = 1, \ldots, c$ when all likelihoods have distinct values.

The recall rate for each application can be computed as follows:

$$\text{recall}_i = \sum_{j\in\{j:\arg\max_k l_k(O_j)=i\}} p_{ij}. \tag{15}$$

Precision rates can also be computed by considering the abundance proportions of each application.

Disclosure

A preliminary shortened version of this paper has been published in [23] by the same authors. An extensive experimentation has been performed since the publication of the preliminary version and the result is presented in the current paper.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work was supported by Inha University and National Research Foundation of Korea grants funded by the Korean Government (NRF-2012R1A1A200665 and NRF-2013R1A1A2059335).

References

[1] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson, "Offline/realtime traffic classification using semi-supervised learning," *Performance Evaluation*, vol. 64, no. 9–12, pp. 1194–1213, 2007.

[2] G. Munz, H. Dai, L. Braum, and G. Carle, "TCP traffic classification using Markov models," in *Proceedings of the 2nd International Conference on Traffic Monitoring and Analysis (TMA '10)*, pp. 127–140, Zurich, Switzerland, April 2010.

[3] J. Zhang, Y. Xiang, Y. Wang, W. Zhou, Y. Xiang, and Y. Guan, "Network traffic classification using correlation information," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 1, pp. 104–117, 2013.

[4] L. Bernaille, R. Teixeira, and K. Salamatian, "Early application identification," in *Proceedings of the ACM International Conference on Emerging Network Experiments and Technologies (CoNEXT '06)*, Lisbon, Portugal, December 2006.

[5] T. Karagiannis, A. Broido, N. Brownlee, and K. Claffy, "Is P2P dying or just hiding?" in *Proceedings of the 47th Annual IEEE Global Telecommunications Conference (Globecom '04)*, Dallas, Tex, USA, November 2004.

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

[7] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, "Class-of-Service mapping for QoS: a statistical signature-based approach to IP traffic classification," in *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC '04)*, pp. 135–148, ACM, Sicily, Italy, October 2004.

[8] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," in *Proceedings of ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '05)*, pp. 50–60, ACM, Banff, Canada, June 2005.

[9] M. Crotti, M. Dusi, F. Gringoli, and M. Salgarelli, "Traffic classification through simple statistical fingerprinting," *ACM*

SIGCOMM Computer Communication Review, vol. 37, no. 1, pp. 5–16, 2007.

[10] T. Nguyen and G. Armitage, "Training on multiple sub-ows to optimize the use of Machine Learning classifiers in real-world IP networks," in *Proceedings of the 31st IEEE Conference on Local Computer Networks*, Tampa, Fla, USA, November 2006.

[11] A. Castiglione, A. De Santis, and F. Palmieri, "Characterizing and classifying card-sharing traffic through wavelet analysis," in *Proceedings of the 3rd International Conference on Intelligent Networking and Collaborative Systems (INCoS '11)*, pp. 691–697, IEEE, Fukuoka, Japan, December 2011.

[12] F. Palmieri, U. Fiore, A. Castiglione, and A. de Santis, "On the detection of card-sharing traffic through wavelet analysis and Support Vector Machines," *Applied Soft Computing Journal*, vol. 13, no. 1, pp. 615–627, 2013.

[13] F. Palmieri, U. Fiore, and A. Castiglione, "A distributed approach to network anomaly detection based on independent component analysis," *Concurrency Computation: Practice and Experience*, vol. 26, no. 5, pp. 1113–1129, 2014.

[14] U. Fiore, F. Palmieri, A. Castiglione, and A. D. Santis, "Netowrk anomaly detection with the restricted Boltzmann machine," *Nuerocomputing*, vol. 122, pp. 13–23, 2013.

[15] Z. Chen, B. Yang, Y. Chen, A. Abraham, C. Grosan, and L. Peng, "Online hybrid traffic classifier for Peer-to-Peer systems based on network processors," *Applied Soft Computing*, vol. 9, no. 2, pp. 685–694, 2009.

[16] W. Ye and K. Cho, "Hybrid P2P traffic classification with heuristic rules and machine learning," *Soft Computing*, vol. 18, no. 9, pp. 1815–1827, 2014.

[17] F. Palmieri and U. Fiore, "A nonlinear, recurrence-based approach to traffic classification," *Computer Networks*, vol. 53, no. 6, pp. 761–773, 2009.

[18] A. Dainotti, W. D. Donato, A. Pescapè, and P. S. Rossi, "Classification of network traffic via packet-level hidden Markov models," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '08)*, pp. 1–5, IEEE, New Orleans, La, USA, December 2008.

[19] X. Mu and W. Wu, "A parallelized network traffic classification based on hidden Markov model," in *Proceedings of the International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC '11)*, pp. 107–112, IEEE, Beijing, China, October 2011.

[20] T. Cover and J. Thomas, *Elements of Information Theory*, John Wiley & Sons, New York, NY, USA, 1991.

[21] H. Akaike, "Information theory and an extension of the maximum like-lihood principle," in *Proceedings of the 2nd International Symposium on Information Theory*, pp. 267–281, Budapest, Hungary, 1973.

[22] SimpleWeb Traces, http://www.simpleweb.org/wiki/Traces.

[23] J. Kim, J. Hwang, and K. Kim, "Internet traffic classification using a Markov model and Kullback-Leibler divergence," in *Proceedings of the International Conference on Communication and Mobile Technology*, July 2013.