TCP Traffic Classification Using Markov Models

Gerhard Münz[1], Hui Dai[2], Lothar Braun[1], Georg Carle[1]

Network Architectures and Services – Institute for Informatics
Technische Universität München, Germany
[1]{muenz|braun|carle}@net.in.tum.de, [2]dai@in.tum.de

Abstract. This paper presents a novel traffic classification approach which classifies TCP connections with help of observable Markov models. As traffic properties, payload length, direction, and position of the first packets of a TCP connection are considered. We evaluate the accuracy of the classification approach with help of packet traces captured in a real network, achieving higher accuracies than the cluster-based classification approach of Bernaille [1]. As another advantage, the complexity of the proposed Markov classifier is low for both training and classification. Furthermore, the classification approach provides a certain level of robustness against changed usage of applications.

1 Introduction

Network operators are interested in identifying the traffic of different applications in order to monitor and control the utilization of the available network resources. Since the traffic of many new applications cannot be identified by specific port numbers, deep packet inspection (DPI) is the current technology of choice. However, DPI is very costly as it requires a lot of computational resources as well as up-to-date signatures of all relevant applications. Furthermore, DPI is limited to unencrypted traffic. Therefore, traffic classification using statistical methods has become an important area of research.

In this paper, we present a novel classification approach which models transitions between data packets using Markov models. While most existing Markov-based traffic classification methods rely on hidden Markov models (HMMs), we make use of observable Markov models where each state directly reflects certain packet attributes, such as the payload length, the packet direction, and the position within the connection. Using training data, separate Markov models are estimated for those applications which we want to identify and distinguish. The classification of new connections is based on the method of maximum likelihood which selects the application whose Markov model yields the highest a-posteriori probability for the given packet sequence.

We restrict the evaluation of our approach to the classification of TCP traffic. Based on traffic traces captured in our department network, we compare the outcome of the Markov classifier with the results of Bernaille's cluster-based classification approach [1]. Furthermore, we show an example of changed application usage and its effect on the classification accuracy. Last but not least, we assess and discuss the complexity of the Markov classifier.

After giving an overview on existing Markov-based traffic classification approaches in Section 2, we explain our approach in Section 3. Section 4 presents the evaluation results before Section 5 concludes this paper.

2 Related Work

In the recent past, various research groups have proposed the utilization of statistical methods for traffic classification. Common to these approaches is that application specific traffic characteristics are learned from training data. Typically, the considered properties are statistics derived from entire flows or connections, or attributes of individual packets. Examples for these two kinds of properties are the average packet length and the length of the first packet of a connection, respectively. Nguyen and Armitage provide a comparison of various existing approaches in a survey paper [2]. In the following, we give an overview on existing traffic classification approaches which make use of Markov models.

Wright et al. [3] and Dainotti et al. [4] estimate a separate HMM for each application considering packet lengths and inter-arrival times. In an HMM, the output of a state is not deterministic but randomly distributed according to the emission probability distribution of the state. While the state output is observable, transitions between states are hidden. Readers looking for a comprehensive introduction to HMMs are referred to Rabiner's tutorial [5]. Wright [3] considers TCP connections and deploys left-right HMMs with a large number of states and discrete emission probability distributions. In contrast, Dainotti [4] generates ergodic HMMs with four to seven states and Gamma-distributed emission probabilities for unidirectional TCP and UDP traffic from clients to servers; packets without payload are ignored. In both cases, traffic classification assigns new connections to the application whose HMM yields the maximum likelihood.

Our approach is motivated by Estevez-Tapiador et al. who use observable ergodic Markov models for detecting anomalies in TCP connections [6]. In an observable Markov model, each state emits a different symbol, which allows deducing the state transitions from a series of observations directly. In the case of Estevez-Tapiador et al., the Markov model generates the sequence of TCP flag combinations observed in those packets of a TCP connection which are sent from the client to the server. Hence, every state represents a specific combination of TCP flags, every transition the arrival of a new packet in the same TCP connection. The transition matrix is estimated using training data which is free of anomalies. During the detection phase, anomalies are then detected by calculating the a-posteriori probability and comparing it with a lower threshold. Estevez-Tapiador et al. use separate Markov models for different applications which are distinguished by their well-known port numbers.

We adopt and extend the modeling approach of Estevez-Tapiador for classifying TCP connections. The training phase is identical: we estimate distinct Markov models for the different applications using training data. In the classification phase, however, we calculate the a-posteriori probabilities of an observed connection for all Markov models. Thereafter, the connection is assigned to the application for which the Markov model yields the maximum a-posteriori probability. In contrast to Estevez-Tapiador, we consider both directions of the TCP connection and take payload lengths instead of TCP flag combinations into account.

In prior work [7], we achieved good classification results with states reflecting the payload length and PUSH flag of each packet. However, the deployed Markov models did not consider the position of each packet within the connection although the packet position strongly influences the payload length distribution and the occurrence probability of the PUSH flag. In this paper, we present a new variant of the Markov classifier which is based on left-right Markov models instead of ergodic Markov Models. Hence, we are able to incorporate the dependency of transition probabilities on the packet's position within the TCP connection. The next section explains this approach in more detail.

3 TCP Traffic Classification Using Markov Models

Just like other statistical traffic classification approaches, we assume that the communication behavior of an application influences the resulting traffic. Hence, by observing characteristic traffic properties, it should be possible to distinguish applications with different behaviors. One such characteristic property is the sequence of packet lengths observed within a flow or connection, which serves as input to many existing traffic classification methods [1, 3, 4].

We use observable Markov models to describe the dependencies between subsequent packets of a TCP connection. The considered packet attributes are payload lengths (equaling the TCP segment size), packet direction, and packet position within the connection. Considering the TCP payload length instead of the IP packet length has the advantage that the value is independent of any IP and TCP options. Similar to several existing approaches (e.g., [1, 4]), we only take into account packets carrying payload. We call these packets "data packets" in the following. The reason for ignoring empty packets is that these are either part of the three-way handshake, which is common to all TCP connections, or they represent acknowledgments. In both cases, the packet transmission is mainly controlled by the transport layer and not by the application. The packet direction denotes whether the packet is sent from the client to the server or vice versa. As client, we always consider the host which initiates a TCP connection.

In contrast to our previous work [7], we do not consider any TCP flags although the occurrence of the PUSH flag may be influenced by how the application passes data to the transport layer. However, an experimental evaluation and comparison of TCP implementations showed that the usage of the PUSH flag varies a lot between different operating systems. Hence, slight improvements of the classification results which can be achieved by considering the PUSH flag might not be reproducible if other operating systems are deployed. As another difference to our previous work, we take into account the packet position within the TCP connection. This leads to better models since the probability distribution of payload length and direction typically depends on the packet position, especially in the case of the first packets of the connection. Moreover, the classification accuracy can be increased because payload length and direction at specific packet positions are often very characteristic for an application. For example, the majority of HTTP connections start with a small request packet sent from the client to the server, followed by a longer series of long packets from the server to the client. In Section 4.1, we empirically confirm these assumptions by looking at TCP connections of different applications.

In general, a Markov model consists of n distinct states $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$, a vector of initial state probabilities $\Pi = (\pi_1, \ldots, \pi_n)$, and an $n \times n$ transition matrix $A = \{a_{\sigma_i,\sigma_j}\}$. In our case, each state represents a distinct combination of payload length, packet direction, and packet position within the TCP connection. The initial state reflects the properties of the first packet within the TCP connection. A transition from one state to the next state corresponds to the arrival of a new packet. The next state then describes the properties of the new packet.

To obtain a reasonably small number of states, the payload lengths are discretized into a few intervals. We evaluated different interval definitions and found that good classification results can be obtained with a rather small number of intervals. The evaluation results presented in Section 4 are based on the following four intervals: [1,99], [100,299], [300, MSS-1], [MSS]. The value of the maximum sequence size (MSS) is often exchanged in a TCP option during the TCP three-way handshake. Alternatively, MSS can be deduced from the maximum observed payload length unless the connection does not contain any packet of maximum payload length. A fallback option is to set MSS to a reasonable default value. Another measure to keep the number of states small is to limit the Markov model to a maximum of l data packets per TCP connection. Hence, if a connection contains more than l data packets, we only consider the first l of them. In order to find a good value for l, we evaluated different settings and show the classification results for $l = 3, \ldots, 7$ in Section 4.

The initial state and transition probabilities are estimated from training data using the following equations:

$$\pi_{\sigma_i} = \frac{F_0(\sigma_i)}{\sum_{m=1}^{n} F_0(\sigma_m)} \quad ; \quad a_{\sigma_i,\sigma_j} = \frac{F(\sigma_i, \sigma_j)}{\sum_{m=1}^{n} F(\sigma_i, \sigma_m)} \tag{1}$$

$F_0(\sigma_i)$ is the number of initial packets matching the state σ_i. $F(\sigma_i, \sigma_j)$ is the frequency of transitions from packets described by state σ_i to packets described by state σ_j. Since the packet position is reflected in the state definitions, we obtain a left-right Markov model with l stages corresponding to the l first data packets in the TCP connection. In our case, every stage comprises eight states representing four payload length intervals and two directions. An example of such a Markov model with $l = 4$ stages is given in Figure 1. $L = 1, \ldots, 4$ denote the different payload length intervals, $C \Rightarrow S$ and $S \Rightarrow C$ the two directions from client to server and server to client.

Only transitions from one stage to the next (left to right) may occur, which means that at most $8^2(l-1)$ out of $(8l)^2$ transition matrix elements are nonzero. Apart from the packet position, the states within each of the stages describe

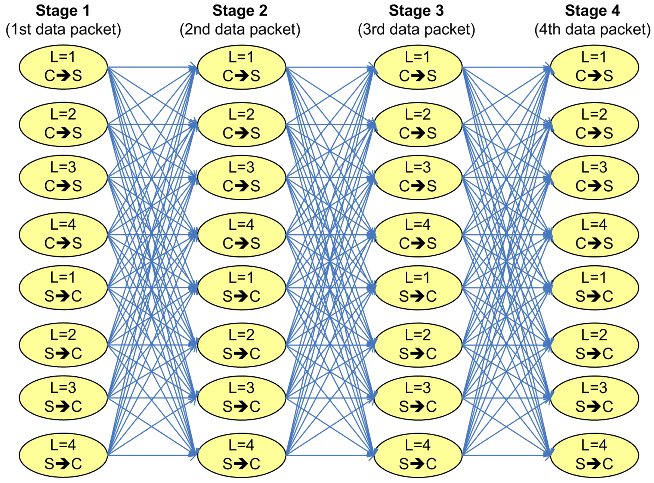

Fig. 1. Left-right Markov model

the same set of packet properties. Therefore, we may alternatively interpret the model as a Markov model with eight states and a time-variant 8×8 transition matrix $A_t, t = 1, \ldots, (l-1)$. This interpretation enables a much more memory efficient storage of the transition probabilities than one large $8l \times 8l$ matrix.

For every application k, we determine a separate Markov model $M^{(k)}$. For this purpose, the training data must be labeled, which means that every connection must be assigned to one of the applications. In order to obtain reliable estimates of the initial and transition probabilities, the training data must contain a sufficiently large number of TCP connections for each application. On the other hand, it is not necessary that all connections contain at least l data packets since the estimation does not require a constant number of observations for every transition. Instead of individual applications, we may also use a single Markov model for a whole class of applications. This approach is useful if multiple applications are expected to show a similar communication behavior, for example because they use the same protocol.

Figure 2 illustrates how the resulting Markov models are used to classify new TCP connections. Given the first l packets of a TCP connection $O = \{o_1, o_2, \ldots, o_l\}$, the log-likelihood for this observation is calculated for all Markov models $M^{(k)}$ with $\Pi^{(k)} = (\pi_1^{(k)}, \ldots, \pi_n^{(k)})$ and $A^{(k)} = \{a_{\sigma_i,\sigma_j}^{(k)}\}$ using the following equation:

$$\log \Pr\left(O|M^{(k)}\right) = \log\left(\pi_{o_1}^{(k)} \prod_{i=1}^{l-1} a_{o_i,o_{i+1}}^{(k)}\right) = \log \pi_{o_1}^{(k)} + \sum_{i=1}^{l-1} \log a_{o_i,o_{i+1}}^{(k)} \qquad (2)$$

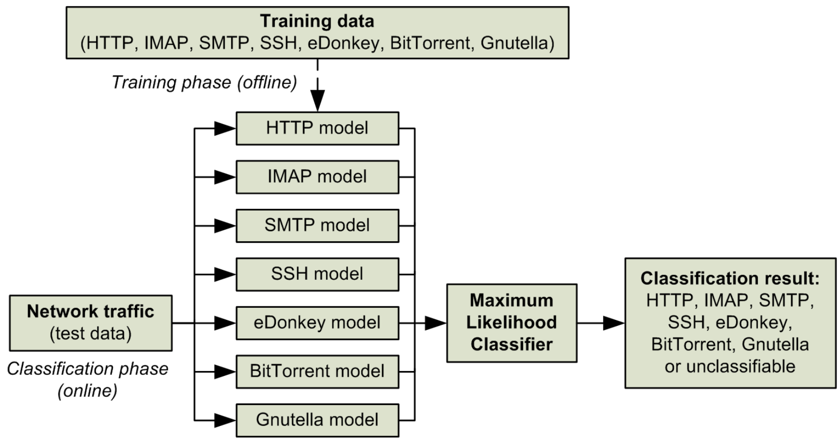

Fig. 2. Traffic classification using Markov models

The maximum likelihood classifier then selects the application for which the log-likelihood is the largest. If a connection contains less than l data packets, the log-likelihood is calculated for the available number of transitions only.

It is possible that a TCP connection to be classified contains an initial state for which $\pi_{o_1}^{(k)} = 0$, or a transition for which $a_{o_i,o_{i+1}}^{(k)} = 0$. This means that such an initial state or transition has not been observed in the training data. Thus, the connection does not fit to the corresponding Markov model. Furthermore, if an unknown initial state or transition occurs in every model, the connection cannot be assigned to any application. This approach, however, may lead to unwanted disqualifications if the training data does not cover all possible traffic, including very rare transitions.

As the completeness of the training data usually cannot be guaranteed, we tolerate a certain amount of non-conformance but punish it with a very low likelihood. For this purpose, we replace all $\pi_{\sigma_i}^{(k)} = 0$ and all $a_{\sigma_i,\sigma_j}^{(k)} = 0$ by a positive value ϵ which is much smaller than any of the estimated non-zero probabilities. Then, we reduce the remaining probabilities to ensure $\sum_i \pi_{\sigma_i}^{(k)} = \sum_j a_{\sigma_i,\sigma_j}^{(k)} = 1$. In the evaluation in Section 4, we use $\epsilon = 10^{-5} = 0.001\%$, which is very small compared to the smallest possible estimated probability of $\frac{1}{300} = 0.33\%$ (300 is the number of connections per application in the training data).

Despite of the uncertainty regarding the completeness of the training data, we want to limit the number of tolerated ϵ-states and ϵ-transitions per connection. This is achieved by setting a lower threshold of $3 \log \epsilon$ for the log-likelihood, which corresponds to three unknown transitions, or an unknown initial state plus two unknown transitions. Connections with a log-likelihood below this threshold are considered unclassifiable.

4 Evaluation

4.1 Training and Test Data

We evaluated the presented traffic classification approach using TCP traffic traces captured in our department network. The traces comprise four classical client-server applications (HTTP, IMAP, SMTP, and SSH) and three peer-to-peer (P2P) applications (eDonkey, BitTorrent, and Gnutella). An accurate assignment of each TCP connection to one of the applications is possible as the HTTP, IMAP, SMTP, and SSH traffic involved our own servers. The P2P traffic, on the other hand, originated or terminated at hosts on which we had installed the corresponding peer-to-peer software; no other network service was running.

The training data consists of 300 TCP connections of each application. The evaluation of the classification approach is based on test data containing 500 connections for each application. In order to enable a comparison with the cluster-based classification approach by Bernaille [1], we only consider connections with at least four data packets. In principle, our approach also works for connections with a smaller number of data packets, yet the classification accuracy is likely to decreases in this case.

Using boxplots, Figure 3 illustrates the payload length distribution of the first seven data packets in the TCP connections contained in the training data. The packet direction is encoded in the sign: payload lengths of packets sent by the server are accounted with a negative sign. In addition to the seven applications used for classification, there is another boxplot for HTTP connections carrying Adobe Flash video content which will be discussed later in Section 4.5. The upper and lower end of the boxes correspond to the 25% and 75% quantiles, the horizontal lines in the boxes indicate the medians. The length of the whiskers is 1.5 times the distance between 25% and 75% quantile. Crosses mark outliers.

As can be seen, two groups of protocols can be distinguished by looking at the first data packet. In the case of SMTP and SSH, the server sends the first data packet, in all other cases, it is the client. Protocols, such as IMAP or SMTP, which specify a dialog in which client and server negotiate certain parameters, are characterized by alternating packet directions. In contrast, the majority of the HTTP connections follow a simple scheme of one short client request followed by a series of large packets returned by the server.

4.2 Evaluation Metrics

As evaluation metrics, we calculate *recall* and *precision* for every application k:

$$recall_k = \frac{\text{number of connections correctly classified as application } k}{\text{number of connections of application } k \text{ in the test data}}$$

$$precision_k = \frac{\text{number of connections correctly classified as application } k}{\text{total number of connections classified as application } k}$$

These two metrics are frequently used for evaluating statistical classifiers. A perfect classifier achieves 100% recall and precision for all applications. Recall is

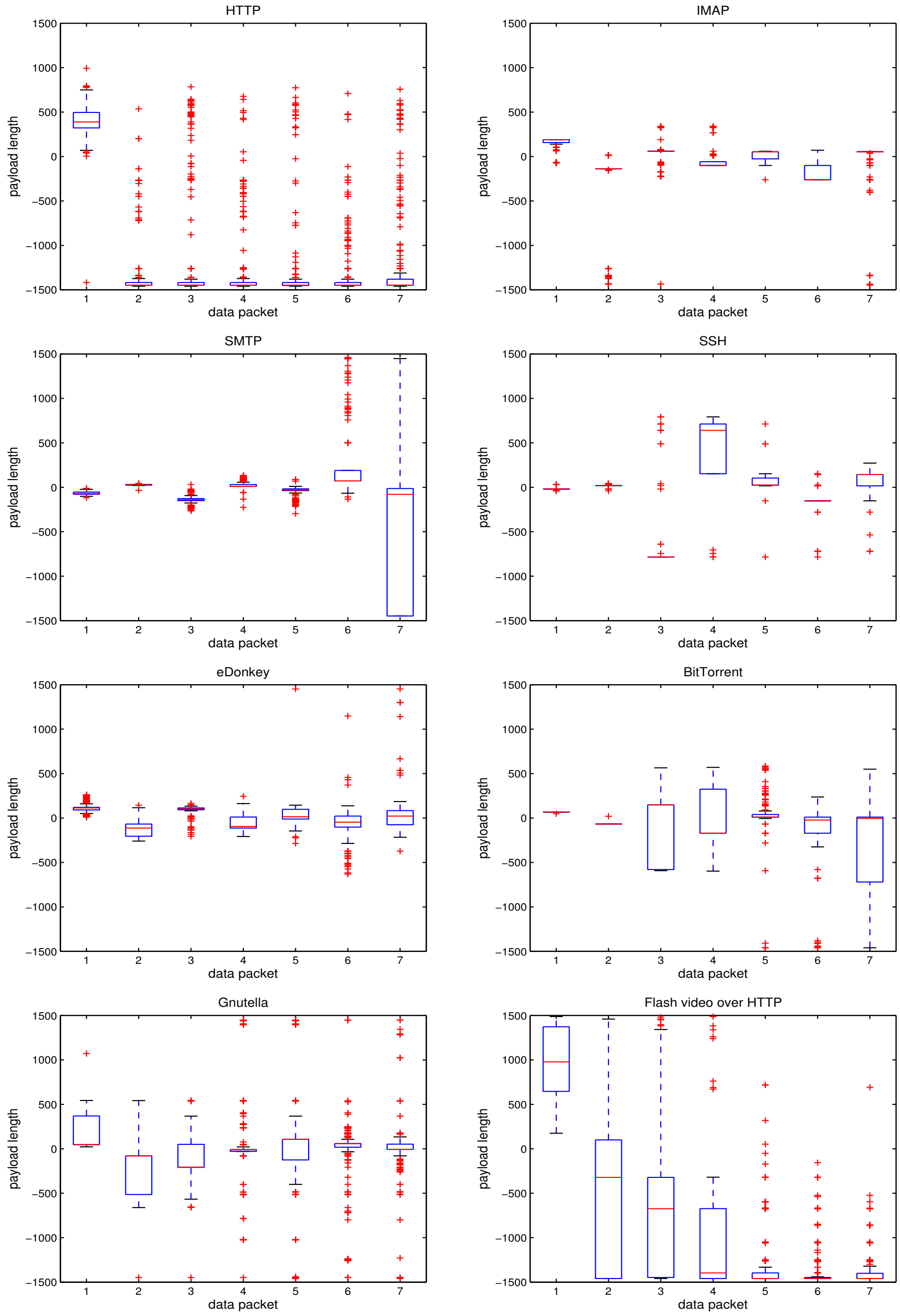

Fig. 3. Payload lengths of first data packets

Table 1. Classification results of Markov classifier

	3 stages			4 stages			5 stages		
	Recall	Prec.	Uncl.	Recall	Prec.	Uncl.	Recall	Prec.	Uncl.
HTTP	96.00%	97.17%	0.00%	98.80%	95.92%	0.00%	97.20%	97.59%	0.00%
IMAP	94.60%	75.20%	0.00%	94.80%	97.33%	0.00%	95.00%	97.94%	0.20%
SMTP	99.60%	94.86%	0.00%	99.80%	95.23%	0.00%	99.80%	95.23%	0.00%
SSH	99.00%	99.80%	0.00%	99.20%	99.60%	0.00%	99.20%	99.80%	0.20%
eDonkey	55.00%	99.28%	0.00%	87.20%	99.09%	0.00%	89.00%	99.55%	0.00%
BitTorrent	98.80%	86.67%	0.00%	98.80%	89.98%	0.00%	99.40%	91.03%	0.00%
Gnutella	97.20%	95.48%	0.00%	95.40%	97.95%	0.00%	97.20%	97.01%	0.00%
Average	91.46%	92.64%	0.00%	96.29%	96.44%	0.00%	96.69%	96.88%	0.06%

	6 stages			7 stages		
	Recall	Prec.	Uncl.	Recall	Prec.	Uncl.
HTTP	97.20%	97.79%	0.20%	97.20%	98.38%	0.20%
IMAP	94.80%	99.79%	0.40%	94.80%	99.79%	0.40%
SMTP	99.60%	95.40%	0.20%	99.60%	95.40%	0.20%
SSH	99.40%	99.40%	0.20%	99.40%	100%	0.40%
eDonkey	93.80%	99.79%	0.00%	97.40%	99.19%	0.00%
BitTorrent	98.40%	93.18%	0.20%	98.40%	96.85%	0.20%
Gnutella	96.60%	96.60%	0.40%	96.80%	96.61%	1.00%
Average	97.11%	97.42%	0.23%	97.66%	98.03%	0.34%

independent of the traffic composition, which means that it does not matter how many connections of the test data belong to application k. In contrast, precision depends on the traffic composition in the test data since the denominator usually increases for larger numbers of connections not belonging to application k. Using test data which contains an equal number of connections for every application, we ensure that the calculated precision values are unbiased.

In order to compare different classifiers with a single value, we calculate the *overall accuracy*, which is usually defined as the number of correctly classified connections divided by the total number of connections in the test data. Since the number of connections per application is constant in our case, the overall accuracy is identical to the average recall. Note that the accuracy values mentioned in this document cannot be directly compared to accuracies mentioned in many related publications which are usually based the unbalanced traffic compositions observed in real networks.

4.3 Classification Results

Table 1 shows the classification results for different numbers of stages l. In addition to recall and precision, the table indicates the percentage of unclassifiable connections for every application. These connections could not be assigned to any application because the maximum log-likelihood is smaller than the lower threshold $3 \log 10^{-5} = -15$. As explained in Section 3, we apply this threshold to sort out connections which differ very much from all Markov models.

As can be seen in the table, the recall values of most applications increase or do not change much if the Markov models contain more stages, which means that more transitions between data packets are considered. Stage $l = 5$ is an exception because HTTP reaches a much higher and Gnutella a much lower recall value than for the other setups. We inspected this special case and saw that 11 to 13 Gnutella connections are usually misclassified as HTTP traffic and vice versa. If the Markov models contain four stages, however, 21 Gnutella connections are misclassified as HTTP, and only four HTTP connections are misclassified as Gnutella, which leads to unusual recall (and precision) values.

Except for Markov models with seven stages, eDonkey is the application with the largest number of misclassified connections. In fact, a large number of eDonkey connections are misclassified as BitTorrent and IMAP traffic. For example, in the case of four stages, 53 eDonkey connections are assigned to BitTorrent, another 11 eDonkey connections to IMAP. These numbers decrease with larger numbers of stages. The example of eDonkey nicely illustrates the relationship between a low recall value for one application and low precision values for other applications: low recall values of eDonkey coincide with low precision values of BitTorrent and IMAP. The recall value of IMAP stays below 95% because 24 IMAP connections are classified as SMTP in all setups.

The precision values show little variation and increase gradually with larger numbers of stages. Finally, the number of unclassifiable connections increases for larger numbers of stages. The reason is that more transitions are evaluated, which also increases the probability of transitions which did not appear in the training data. Although we account unknown initial states and transitions with ϵ-probability, connections with three or more of these probabilities are sorted out by the given threshold. Obviously, the number of unclassifiable connections could be reduced by tolerating a larger number of unknown transitions. Alternatively, we could increase the number of connections in the training data in order to cover a larger number of rare transitions.

The average recall, which is equal to the overall accuracy, jumps from 91.46% to 96.29% when the number of stages is increased from three to four. At the same time, the average precision increases from 92.64% to 96.44%. Thereafter, both averages increase gradually with every additional stage. Hence, at least four data packets should be considered in the Markov models to obtain good results.

4.4 Comparison with Bernaille's Approach

Bernaille [1] proposed a traffic classification method which uses clustering algorithms to find connections with similar payload lengths and directions in the first data packets. The Matlab code of this method can be downloaded from a website [8]. Bernailles's approach requires that all connections in the test and training data have at least as many data packets as analyzed by the classification method. Furthermore, the results of his work show that best results can be achieved with three or four data packets. As mentioned in Section 4.1, we

Table 2. Classification results of Bernaille's classifier

	3 data packets, 27 clusters		4 data packets, 34 clusters		3 data packets, 28 clusters		3 data packets, 29 clusters	
	Recall	Prec.	Recall	Prec.	Recall	Prec.	Recall	Prec.
HTTP	88.60%	96.30%	99.60%	86.16%	88.00%	94.62%	90.20%	95.35%
IMAP	91.00%	96.19%	93.20%	99.79%	92.60%	83.88%	87.80%	90.89%
SMTP	98.80%	95.18%	97.40%	95.49%	90.20%	100%	98.80%	95.37%
SSH	97.20%	98.98%	95.40%	99.58%	97.40%	100%	97.80%	98.79%
eDonkey	95.80%	87.09%	98.80%	98.80%	91.00%	92.11%	100%	89.61%
BitTorrent	88.80%	100%	93.60%	100%	96.80%	95.09%	97.20%	100%
Gnutella	96.40%	85.61%	92.00%	92.37%	96.20%	88.75%	95.20%	97.74%
Average	93.80%	94.19%	95.71%	96.03%	93.17%	93.49%	95.29%	95.39%

prepared our datasets for a comparison with Bernaille by including connections with at least four data packets only.

The learning phase of Bernaille's classifier is nondeterministic and depends on random initialization of the cluster centroids. Furthermore, the number of clusters as well as the number of data packets needs to be given as input parameters to the training algorithm. The documentation of Bernaille's Matlab code recommends 30 to 40 clusters and three to four data packets as a good start point. At the end of the clustering, the algorithm automatically removes clusters which are assigned less then three connections of the training data. A calibration method performs the training of the classifier with different numbers of clusters and data packets and returns the model which achieves the highest classification accuracy with respect to the training data.

As recommended by Bernaille, we ran the calibration method to cluster the connections in the training data with 30, 35, and 40 initial cluster centroids and three and four data packets. The best classifier was then used to classify the test data by assigning each of the connections to the nearest cluster. Further improvements, which Bernaille achieved by considering port numbers in addition to cluster assignments [1], were not considered since our approach does not evaluate port numbers either.

Table 2 shows the classification results for four different runs of the calibration method. As can be seen, the average recall and precision values do not reach the same level as the Markov classifier. A possible explanation is that Bernaille's approach does not consider any correlation between subsequent packets. The classification results vary a lot between different runs of the calibration method. Interestingly, we obtain very different results in the third and forth run although both classifiers use three data packets and a very similar number of clusters. The range of the recall values obtained for an individual application can be very wide. The most extreme example is HTTP with recall values ranging from 88.6% to 99.6%. In general, we observed that the classification results depend very much on the initialization values of the cluster centroids and not so much on the remaining parameters, such as the number of clusters and data packets.

Table 3. Classification of Flash over HTTP traffic

	tolerant classifier			intolerant classifier		
	HTTP	Gnutella	Uncl.	HTTP	Gnutella	Uncl.
4 stages	68.0%	17.4%	14.6%	60.0%	11.0%	29.0%
5 stages	63.8%	16.6%	19.6%	44.6%	14.4%	41.0%
6 stages	60.8%	17.0%	22.2%	43.8%	11.6%	44.6%
7 stages	61.2%	16.0%	22.8%	39.6%	11.6%	48.8%

In contrast to Bernaille's approach, the training of the Markov classifier always yields deterministic models which do not depend on any random initialization. Hence, we do not need to run the training method several times, which is an advantage regarding the practical deployment.

4.5 Change of Application Usage

HTTP has become a universal protocol for various kinds of data transports. Many websites now include multimedia contents, such as animated pictures or videos. There are many sites delivering such contents, with *www.youtube.com* being one of the most popular. A large proportion of these embedded multimedia contents are based on Adobe Flash. Flash typically transfers data in streaming mode, which means that after a short prefetching delay the user can start watching the video without having to wait until the download is finished.

In order to assess how our classification approach behaves if the usage of an application changes, we applied the classifier to 500 HTTP connections carrying Flash video content. These connections were captured in our university network and identified by the HTTP content type "video/x-flv". The boxplots at the bottom right of Figure 3 show the payload length distribution. Compared to the previously analyzed HTTP connections, which did not include any Flash video downloads, the variance in the first four packets is much larger. The request packets sent from the client to the server tend to contain more payload than in the case of other HTTP traffic whereas the second and third packets are often smaller.

Traffic classification should be robust against such changes of application usage. In the optimal case, the classifier still classifies the corresponding connections correctly as HTTP traffic. Apart from that, it is also acceptable to label the connections as unclassifiable. On the other hand, the connections should not be assigned to wrong applications.

Table 3 shows how the HTTP connections containing Flash video content are classified in dependence of the number of stages in the Markov models. Apart from tolerant classification with $\epsilon = 10^{-5}$ and log-likelihood threshold -15, we tested an intolerant classifier which disqualifies all connections with unknown initial state or transition. The tolerant classifier assigns 60% of the connections to HTTP and around 17% to Gnutella. Hence again, similarities between HTTP and Gnutella traffic cause a certain number of misclassified connections. The remaining connections remain unclassified because the maximum log-likelihood is smaller than $3 \log 10^{-5}$. With the intolerant classifier, twice as many connections remain unclassified, mainly account of connections previously assigned to HTTP. This shows that tolerance of non-conforming connections increases the robustness of the classifier against usage changes.

Although the tolerant classifier still classifies most of the connections as HTTP traffic, the classification accuracy is degraded. To solve this problem, it suffices to re-estimate the Markov model of HTTP with training data covering the new kind of HTTP traffic. Alternatively, we can add a Markov model which explicitly models Flash over HTTP.

4.6 Complexity

The estimation of initial state and transition probabilities using equations (1) requires counting the frequency of initial packet properties and transitions. If the training data contains C connections of an application, estimating the parameters of the corresponding Markov model with l stages requires at most $C \cdot l$ counter increments plus $7 + 56(l-1)$ additions and $8l$ divisions.

In order to classify a connection, the log-likelihood needs to be calculated for every Markov model using equation (2). This calculation requires $(N-1)$ additions, N being the number of analyzed data packets in the connection. The number of stages l is an upper bound for N. The maximum log-likelihood of all Markov models needs to be determined and checked against the given lower threshold. Hence, for K different applications, we have at most $K(l-1)$ additions and K comparisons.

Other statistical traffic classification approaches typically require more complex calculations. This is particularily true for HMMs where emission probabilities have to be considered in addition to transition probabilities. Regarding Bernaille's approach, the clustering algorithm determines the assignment probability of every connection in the training data to every cluster. After recalculating the cluster centroids, the procedure is repeated in another iteration. Just one of these iterations is more complex than estimating the Markov models. The classification of a connection requires calculating the assignment probabilities for every cluster. If Gaussian mixture models (GMMs) are used as in Bernaille's Matlab code, the probabilities are determined under the assumption of multivariate normal distributions, which is more costly than calculating the Markov likelihoods.

5 Conclusion

We presented a novel traffic classification approach based on observable Markov models and evaluated the classification accuracy with help of TCP traffic traces of different applications. The results show that our approach yields slightly better results than Bernaille's cluster-based classification method [1]. Furthermore, it provides a certain level of robustness with respect to the changed usage of an application. After all, the complexity of our approach is low compared to other statistical traffic classification methods.

The classification accuracy depends on the number of stages per Markov model, which corresponds to the maximum number of data packets considered per TCP connection. Based on our evaluation, we recommend Markov models with at least four stages, corresponding to 32 states. Every additional stage gradually improves the accuracy. As an important property, connections whose number of data packets is smaller than the number of stages can still be classified. Hence, the only drawback of maintaining more stages is that more transition probabilities need to be estimated, saved, and evaluated per application.

In order to better assess the performance of our classification approach, we intend to apply it to other traffic traces captured in different networks. Beyond that, it will be interesting to consider additional applications since the set of applications regarded in our evaluation is very limited, of course. Finally, we think of extending the approach to the classification of UDP traffic, which is mainly used for real-time applications.

Acknowledgments

We gratefully acknowledge support from the German Research Foundation (DFG) funding the LUPUS project in which this research work as been conducted.

References

1. Bernaille, L., Teixeira, R., Salamatian, K.: Early application identification. In: Proc. of ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT) 2006, Lisboa, Portugal (2006)
2. Nguyen, T.T.T., Armitage, G.: A survey of techniques for internet traffic classification using machine learning. IEEE Communications Surveys & Tutorials **10** (2008) 56–76
3. Wright, C., Monrose, F., Masson, G.: HMM profiles for network traffic classification (extended abstract). In: Proc. of Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC), Fairfax, VA, USA (2004) 9–15
4. Dainotti, A., de Donato, W., Pescapè, A., Rossi, P.S.: Classification of network traffic via packet-level hidden markov models. In: Proc. of IEEE Global Telecommunications Conference (GLOBECOM) 2008, New Orleans, LA, USA (2008)
5. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of IEEE **77** (1989) 257–286
6. Estevez-Tapiador, J.M., Garcia-Teodoro, P., Diaz-Verdejo, J.E.: Stochastic protocol modeling for anomaly based network intrusion detection. In: Proc. of IEEE International Workshop on Information Assurance (IWIA). (2003)
7. Dai, H., Münz, G., Braun, L., Carle, G.: TCP-Verkehrsklassifizierung mit Markov-Modellen. In: Leistungs-, Zuverlässigkeits- und Verlässlichkeitsbewertung von Kommunikationsnetzen und Verteilten Systemen, 5. GI/ITG-Workshop MMBnet 2009, Hamburg, Germany (2009)
8. Bernaille, L.: Homepage of early application identification. `http://www-rp.lip6.fr/~teixeira/bernaill/earlyclassif.html` (2009)