# Improving Markov-based TCP Traffic Classification

**Gerhard Münz[1], Stephan Heckmüller[2], Lothar Braun[1], and Georg Carle[1]**

1 **Network Architectures and Services, Technische Universität München, Germany**
{muenz|braun|carle}@net.in.tum.de
2 **Telecommunications and Computer Networks, Universität Hamburg, Germany**
heckmueller@informatik.uni-hamburg.de

**Abstract**

This paper presents an improved variant of our Markov-based TCP traffic classifier and demonstrates its performance using traffic captured in a university network. Payload length, flow direction, and position of the first data packets of a TCP connection are reflected in the states of the Markov models. In addition, we integrate a new "end of connection" state to further improve the classification accuracy. Using 10-fold cross validation, we identify appropriate settings for the payload length intervals and the number of data packets considered in the models. Finally, we discuss the classification results for the different applications.

## 1 Introduction

Traffic classification based on port numbers has reached its limits since a one-to-one mapping between port and application does not exist for many modern network services. Examples are peer-to-peer applications, which often allocate ports dynamically, and web applications, such as streaming servers, which run on top of HTTP on the same port 80 or 443. Classification based on deep packet inspection (DPI) involves pattern matching on packet payload which is computationally intensive. Moreover, DPI is restricted to unencrypted traffic. Therefore, network operators are interested in statistical classification methods which consider traffic characteristics beyond packet contents.

The common assumption of statistical traffic classification is that the communication behavior of an application influences the resulting traffic. Hence, by observing appropriate traffic properties, it should be possible to distinguish applications with different behaviors. In the literature, the application of machine learning techniques to the traffic classification problem is very popular. Examples can be found in the survey paper of Nguyen and Armitage [8]. Similarly, multiple approaches to traffic classification are evaluated and compared in the paper of Kim et al. [6]. Among other approaches, the authors consider various machine learning algorithms with Support Vector Machines being the most accurate. In the article of Este et al. [4], the amount of information carried by various traffic features is investigated. The authors evaluate the *mutual information* of the traffic features and the application type. They conclude that the packet sizes exhibit the highest amount of information concerning the application type for multiple data sets.

In a previous paper, we present a novel TCP traffic classification approach using observable left-right Markov models [7]. This work is motivated by Estevez-Tapiador et al. who deploy observable ergodic Markov models for detecting anomalies in TCP connections [5]. In contrast to Estevez-Tapiador, we do not consider TCP flags but payload length, flow direction, and position of the first data packets of a TCP connection. Our evaluation shows

that taking into account the first four data packets per connection suffices to achieve very good classification results. Evaluating additional data packets allow us to further increases the accuracy. As an advantage to hidden Markov models (HMMs), which have been proposed for traffic classification by Wright et al. [12] and Dainotti et al. [2], the utilization of observable Markov models drastically simplifies the estimation of model parameters as well as the classification of a new connection.

In Section 2 of this paper, we present an improved variant of the Markov-based traffic classifier which uses an additional "end of connection" state to reflect the termination of TCP connections with small numbers of data packets. In Section 3, we apply this approach to TCP traffic which was captured in the Munich Scientific Network and labeled with help of OpenDPI [9], an open-source signature-based traffic classification library. Different parameter settings are compared based on the average accuracy achieved in a 10-fold cross-validation. The most appropriate model parameters are then used to determine the classification results for a realistic traffic mix. All in all, the results confirm earlier findings that our approach yields very high classification accuracies even when dealing with real-world traffic data.

# 2 Markov-based TCP Traffic Classifier

## 2.1 High-Level Description

Our traffic classifier considers the packet sequence at the beginning of a TCP connection as the output of an observable Markov model (or Markov chain). In contrast to HMMs used in other traffic classification approaches [2, 12], each packet in the sequence can be directly linked to a specific state in the Markov model. Hence, given a packet sequence, we can directly infer the initial state from the first packet, as well as all further state transitions from the following packets. If the Markov model's initial state and transition probabilities are known, we can calculate the likelihood of the packet sequence.

Our TCP traffic classifier maintains multiple Markov models with identical states but different initial state and transition probabilities. Each of these models reflects the traffic characteristics of a different application. Given the sequence of the first data packets of a TCP connection, the likelihood is calculated for all available Markov models. The classifier then assigns the TCP connection to the application for which the corresponding Markov model yields the maximum likelihood.

## 2.2 Considered Packet Attributes

The accuracy of the classification depends very much on the considered packet attributes. As a simple example, we could take into account the flow direction of the packets. There are only two possible directions, namely from the client which initiates the TCP connection to server and vice versa. Hence, we can model this attribute in a Markov model with only two states. In fact, the flow direction of the packets is an important aspect of the communication pattern of an application. Nevertheless, this packet attribute alone is not sufficient to accurately distinguish different applications.

In addition to the flow direction, we experimented with further packet attributes, such as the packet length and the TCP flags, and compared the resulting classification accuracies [1]. We found that excellent classification results can be achieved if payload length (equaling the TCP segment size), direction, and position of the first data packets within a TCP connection are considered in the states of the Markov model [7]. Data packets are those packets which

contain at least one byte of payload. All other packets (such as handshake packets and empty acknowledgment packets) are ignored because their transmission is usually triggered by the TCP layer and not by the application.

The payload length of the data packets is quantized into a few intervals in order to map the packet attributes to a reasonably small number of states. We denote these intervals as sets of interval boundaries, i.e. $B_i = \{b_1, \ldots, b_m\}$. Boundary sets with $m$ members lead to a mapping of the packet lengths onto $m+1$ intervals: $[1; b_1], [(b_1+1); b_2], \ldots, [(b_m+1); \infty)$. Another measure to keep the number of states small is to restrict the number of data packets considered in the Markov models. The number and boundaries of the payload length intervals as well as the number of data packets need to be set appropriately to obtain good classification results. We discuss these parameterization issues in Section 3.3.

## 2.3 New "end of connection" State

In the following, we present an improved variant of our traffic classifier. In addition to the states reflecting packet attributes, we add a new state which is reached if a connection terminates before the maximum number of data packets considered in the Markov models is transmitted. The classification of such already ended connections is useful for collecting traffic statistics, characterizing network hosts, etc. As we will show in Section 3.3, including the "end of connection" (EOC) state leads to an improvement of the classification accuracy.

The new EOC state is highlighted in Figure 1 which displays the example of a Markov model considering the first four data packets of a connection. In the example, the payload length is quantized to four intervals $I_1, \ldots, I_4$; the directions from client to server and server to client are indicated as $C \Rightarrow S$ and $S \Rightarrow C$. Hence, the total number of states is 33. Transitions to the EOC state are possible from the first, second, and third data packet.

## 2.4 Model Estimation and Storage

Let's assume that $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ is the set of states of the Markov model. Then, the initial state probabilities can be written as vector $\Pi = (\pi_{\sigma_1}, \ldots, \pi_{\sigma_n})$ where $\pi_{\sigma_i}$ is the probability that the first packet has the attributes described by $\sigma_i$. The transition probabilities form an $n \times n$ matrix $A = \{a_{\sigma_i,\sigma_j}\}$ where $a_{\sigma_i,\sigma_j}$ specifies the probability of a transition from $\sigma_i$ to $\sigma_j$.

The initial state and transition probabilities are estimated from labeled training data, i.e. TCP connections for which the application is known. For every application, the probabilities are estimated with the following equations:

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

$F_0(\sigma_i)$ is the number of initial packets in the training data matching the attributes defined by $\sigma_i$. $F(\sigma_i,\sigma_k)$ is the frequency of transitions from packets described by $\sigma_i$ to packets described by $\sigma_k$. In order to obtain good estimates, a sufficiently large set of TCP connections is needed which is representative for the traffic to be classified later.

In our case, only states representing data packets at the first position may be initial states. Furthermore, transitions only occur from one data packet to the next or alternatively to the EOC state. Hence, maintaining an $n \times 1$ initial state probability vector $\Pi$ and an $n \times n$ transition matrix $A$ is inefficient because most of the elements are zero probabilities by definition. Due to the regular structure of the Markov model, the probabilities can be stored in a much more memory efficient way. The elements in $\Pi$ can be limited to

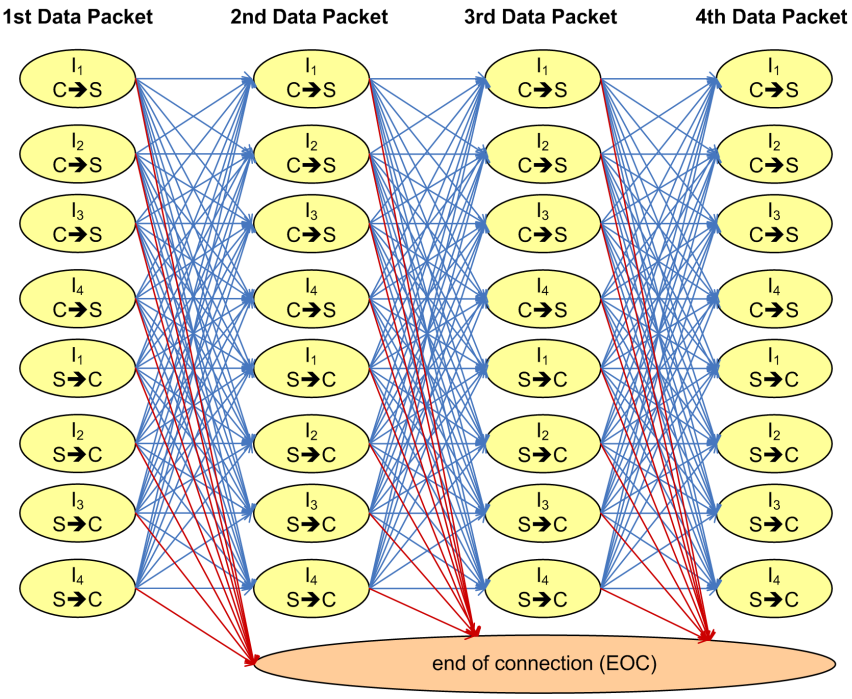


**Figure 1** Markov model of the improved classifier

the states defining attributes of the first data packet. If $l$ is the number of data packets considered, the transition probabilities can be saved in $(l-1)$ small transition matrices $A_t, t = 1, \ldots, (l-1)$, each including only transitions from packet position $t$ to $t+1$. As an example, the probabilities of the Markov model given in Figure 1 can be stored in a $\Pi$ vector of length 8 and three $8 \times 9$ transition matrices $A_t$.

Rare initial states or transitions are often not included in the training data. If such an initial state or transition is observed in a TCP connection to be classified, the likelihood of the Markov model drops to zero and disqualifies the associated application. This may lead to wrong classification results. It may also happen that the connection cannot be assigned to any application because the packet sequence has a zero likelihood for all Markov models. For this reason, we replace all zero probabilities of logically possible initial states and transitions by a positive value $\epsilon$ which is smaller than any of the estimated non-zero probabilities. Afterwards, we normalize the initial state probabilities and the transition probabilities such that $\sum_i \pi_{\sigma_i} = \sum_j a_{\sigma_i,\sigma_j} = 1$ for all models.

If a TCP connection can only be classified because of multiple zero probabilities replaced by $\epsilon$, it does not really seem to match the traffic characteristics of that application. Therefore, we set a lower threshold $h$ for the model likelihood which corresponds to a small number of $\epsilon$ transitions. If this threshold is not exceeded for any application, the connection remains unclassified. Throughout this paper, we use $\epsilon = 10^{-5} = 0.001\%$ and $h = \epsilon^3 = 10^{-15}$.

Concerning the complexity of model estimation, the computation of initial state and transition probabilities using equations (1) requires counting the frequency of initial packet attributes and transitions. The computational effort linearly depends on the number of data packets considered and the number of connections in the training data. Compared to other model types, this is very time-efficient.

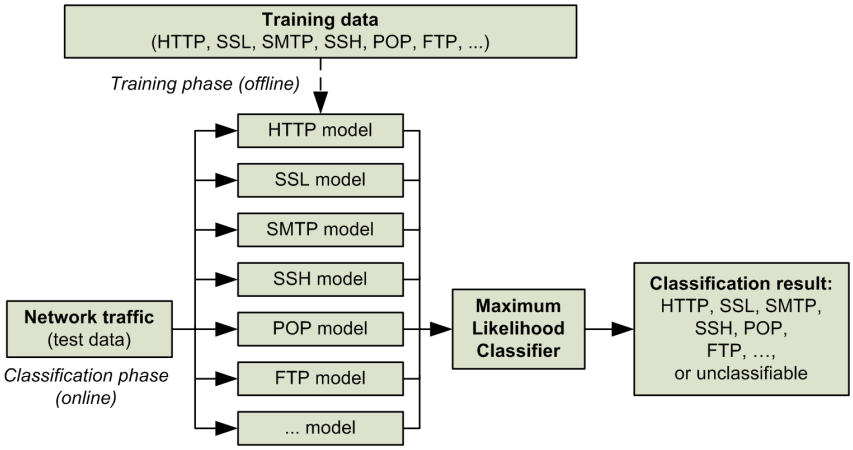


**Figure 2** Traffic classification using Markov models

## 2.5 Classification

During the classification phase, it is beneficial to calculate the log-likelihood instead of the likelihood of the Markov models since this replaces multiplications by additions. Given the first $l$ data packets of a TCP connection $O = \{o_1, o_2, \ldots, o_l\}$, the log-likelihood for this connection is calculated for all Markov models $M^{(k)}$ with $\Pi^{(k)} = (\pi^{(k)}_{\sigma_1}, \ldots, \pi^{(k)}_{\sigma_n})$ and $A^{(k)} = \{a^{(k)}_{\sigma_i,\sigma_j}\}$ using the following equation:

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

If the connection contains less than $l$ data packets (i.e. fewer data packets than considered in the Markov models), the likelihood calculation stops after adding the logarithm of the transition probability to the EOC state. The classifier finally selects the application for which the log-likelihood is the largest. The overall classification process is illustrated in Figure 2. The block diagram shows how the Markov models obtained during the training phase are used to classify new TCP connections.

The computational effort necessary to classify a TCP connection is determined by the computation of the log-likelihood which has to be calculated for every Markov model using equation (2). This effort linearly depends on the number of models and on the length of the state sequence which is bounded by the number of data packets considered in the models.

Other statistical traffic classification methods typically require more complex calculations. This is particularly true for HMMs where emission probabilities have to be considered in addition to transition probabilities.

# 3 Evaluation

## 3.1 Training and Test Data

In order to evaluate our traffic classifier, we monitored traffic at the gateway which connects the Munich Scientific Network to the German National Research and Education Network (DFN). For our evaluation, we captured approximately six hours of IPv4/TCP traffic from

**Table 1** Application labels provided by OpenDPI

| Application label | Connections | Percentage | Distinct server ports |
|---|---|---|---|
| http | 1,136,359 | 64.48% | 81 (mainly ports 80, 8080) |
| ssl | 158,331 | 8.98% | 22 (mainly ports 443, 993, 995) |
| smtp | 150,173 | 8.52% | 5 (mainly ports 25, 587) |
| flash | 92,273 | 5.24% | 6 (mainly ports 1953, 80) |
| ssh | 22,087 | 1.25% | 3 (mainly port 22) |
| http+flash | 20,030 | 1.14% | 4 (mainly port 80) |
| pop | 7,390 | 0.42% | 1 (port 110) |
| ftp | 6,232 | 0.35% | 6,035 (73 connections to port 21) |
| windowsmedia | 4,974 | 0.28% | 1 (port 80) |
| imap | 4,367 | 0.25% | 1 (port 143) |
| dns | 2,499 | 0.14% | 1 (port 53) |
| bittorrent | 1,735 | 0.10% | 115 (e.g. ports 8080, 80, 6879) |
| rdp[a] | 1,047 | 0.06% | 1 (port 3389) |
| other | 4,850 | 0.14% | 38 (mainly 80) |
| unknown | 150,013 | 8.51% | 5295 (e.g. ports 10050, 2703) |

[a] remote desktop protocol

and to the class B network of our computer science department. Traffic was captured in one chunk on a busy weekday. We modified the intrusion detection system Bro [10] to classify the captured packets into different TCP connections and to extract the corresponding sequences of payload length and flow direction tuples. Individual TCP packets not belonging to any valid TCP connection and TCP connections without any data packet were filtered out. In total, we extracted packet attributes of 1,762,360 TCP connections, each containing at least one data packet.

In addition to Bro, we applied OpenDPI [9] to obtain an application label for each TCP connection. Since signatures can only be found in unencrypted traffic, connections of applications running on top of SSL or TLS are identically labeled *ssl*. Similarly, all connections of applications making use of SSH are labeled *ssh*. OpenDPI was applied to all packets, which sometimes led to the situation that the label returned by OpenDPI changed in the middle of the connection. Most frequently, this happened for TCP connections which were first labeled *http* and later *flash* (20,300 occurrences), which we combined in a new label *http+flash*. Other types of label changes were observed for 1,544 connections which we all assigned to the label *other* (cf. below).

Table 1 shows the number of TCP connections and the number of distinct server port numbers per application label. Applications with less than 1000 connections are not shown individually but summed up as *other*. It is not surprising that 64.48% of the TCP connections are labeled *http* because HTTP traffic dominates in the Internet today. Although most connections are directed to well-known server ports, we observe a certain number of connections with non-standard server ports. Traffic labeled *ftp* mostly includes data connections of FTP sessions. These connections are typically established from the FTP server to an ephemeral port at the FTP client, which is accounted as server port (from the TCP perspective) in the table. In addition, a small amount of FTP control connections is present. 8.51% of all connections could not be classified by OpenDPI and are labeled *unknown*. Almost 28% of these connections are established between two identical endpoints, one of them running an

unknown service on port 10050. More than 11% are directed to port 2703 which is commonly used by Razor servers hosting databases with signatures for spam detection [11].

We restrict our classifier to application with more than 1000 connections in our data. For each of these applications, we randomly select 1000 connections as training data. Although there are more than 1000 connections, we do not include a Markov model for *http+flash* because these connections get easily confused with *http*. While the training data contains an equal number of connections for all applications, the test data shall reflect the original traffic mix. Therefore, we select 2% of the connections of each application as test data, making sure that none of the connections is contained in training and test data at the same time.

## 3.2 Evaluation Metrics

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

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

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

A perfect classifier achieves 100% recall and precision values for all applications.

In order to compare different classifiers with a single metric, we calculate the *overall accuracy*, which is the proportion of correctly classified connections. In the cross-validation case (see Section 3.3), the overall accuracy is calculated for an equal number of connections per application, which means that the overall accuracy is identical to the average recall.

## 3.3 Parameterization Using Cross-Validation

The performance of the classifier depends on the parameterization, in particular on the number of considered data packets and the payload length interval boundaries. We use 10-fold cross-validation to assess the influence of different parameter settings. For this, the training data is divided into ten disjoint sets. Then, ten classifiers are trained, each one with a different set left out. For each classifier, we use the remaining fold not used for training as test data. To evaluate the overall performance of a parameter setting, we consider the minimum, mean, and maximum classification accuracy resulting from the ten runs.

As a starting point, we use similar interval boundaries as in our previous work [7], namely $B_1 = \{100, 300, 1450\}$. This means that the following four payload length intervals are considered in the states of the Markov models: $[1; 100]$, $[101; 300]$, $[301; 1450]$, $[1451; \infty)$. A preliminary evaluation of $B_1$ showed deficiencies in the classification of *pop* and *smtp*. By inspection of the applications' payload length distributions, we noticed systematic differences in the payload length of the first data packet: For *smtp*, almost 90% of the first data packets exceed 50 byte whereas, in case of *pop*, less than 20% contain more than 50 byte of payload. Therefore, we add 50 as an additional boundary and evaluate the classifier's performance for $B_2 = \{50, 100, 300, 1450\}$.

The interval boundaries mentioned so far are based on experimental evaluation of the classifier's performance and inspection of payload lengths. Given the huge number of possible boundary combinations, the alternative approach of finding interval boundaries in a systematic and automatic manner is particularly complex. In order to limit the complexity, we use an iterative greedy algorithm which determines the best additional boundary based on an existing set of interval boundaries found in preceding iterations. This is accomplished

**Table 2** Cross-validation results for differing number of packets and boundaries

| Packets | Boundaries | Overall accuracy (*min/mean/max*) | |
|---|---|---|---|
| | | Without EOC state | With EOC state |
| 3 Packets | $B_1 = \{100, 300, 1450\}$ | 0.8433/0.8508/0.8708 | 0.8667/0.8764/0.8883 |
| | $B_2 = \{50, 100, 300, 1450\}$ | 0.8758/0.8938/0.9025 | 0.8967/0.9096/0.9258 |
| | $B_3 = \{40, 190\}$ | 0.8442/0.8544/0.8692 | 0.8433/0.8610/0.8792 |
| | $B_4 = \{40, 190, 380\}$ | 0.8842/0.8995/0.9067 | 0.8933/0.9066/0.9183 |
| | $B_5 = \{40, 190, 380, 620\}$ | 0.9092/0.9187/0.9317 | 0.9192/0.9287/0.9367 |
| 4 Packets | $B_1 = \{100, 300, 1450\}$ | 0.8583/0.8749/0.8975 | 0.8842/0.8993/0.9092 |
| | $B_2 = \{50, 100, 300, 1450\}$ | 0.9158/0.9228/0.9333 | 0.9267/0.9397/0.9525 |
| | $B_3 = \{40, 190\}$ | 0.8483/0.8584/0.8700 | 0.8717/0.8834/0.9000 |
| | $B_4 = \{40, 190, 380\}$ | 0.9000/0.9163/0.9292 | 0.9142/0.9309/0.9425 |
| | $B_5 = \{40, 190, 380, 620\}$ | 0.9208/0.9326/0.9450 | 0.9358/0.9463/0.9592 |
| 5 Packets | $B_1 = \{100, 300, 1450\}$ | 0.8725/0.8866/0.9000 | 0.8950/0.9050/0.9183 |
| | $B_2 = \{50, 100, 300, 1450\}$ | 0.9308/0.9381/0.9458 | 0.9400/0.9469/0.9625 |
| | $B_3 = \{40, 190\}$ | 0.8625/0.8760/0.8858 | 0.8908/0.8977/0.9108 |
| | $B_4 = \{40, 190, 380\}$ | 0.9217/0.9318/0.9392 | 0.9408/0.9497/0.9583 |
| | $B_5 = \{40, 190, 380, 620\}$ | 0.9442/0.9504/0.9567 | 0.9567/0.9635/0.9717 |
| 6 Packets | $B_1 = \{100, 300, 1450\}$ | 0.8817/0.8914/0.9067 | 0.8992/0.9102/0.9225 |
| | $B_2 = \{50, 100, 300, 1450\}$ | 0.9300/0.9407/0.9533 | 0.9400/0.9491/0.9550 |
| | $B_3 = \{40, 190\}$ | 0.8725/0.8814/0.8917 | 0.8900/0.8997/0.9075 |
| | $B_4 = \{40, 190, 380\}$ | 0.9300/0.9393/0.9500 | 0.9400/0.9485/0.9575 |
| | $B_5 = \{40, 190, 380, 620\}$ | 0.9433/0.9503/0.9617 | 0.9517/0.9625/0.9750 |

by analyzing the relation of the application labels and the state sequences of the corresponding connections in the training data. We use a concept from decision tree learning, the *Gini impurity index* which measures how "impure" the assignment of state sequences to application labels is (cf. [3]). As a new boundary, we choose the boundary which results in a maximum reduction of impurity in order to achieve a strong differentiation of the applications with respect to the state sequences. Considering state sequences of four data packets, four iterations of this algorithm lead to the following interval boundary sets: $\{190\}$, $B_3 = \{40, 190\}$, $B_4 = \{40, 190, 380\}$, $B_5 = \{40, 190, 380, 620\}$. We restrict ourselves to models with at least two boundaries which turned out to be the minimum number necessary for achieving acceptable classification results.

In the following experiments, we compare the overall accuracy of five boundaries sets $B_1, \ldots, B_5$. Moreover, we assess the influence of the EOC state. The resulting minimum, mean, and maximum overall accuracies are tabulated in Table 2 for 3 to 6 data packets. As can be seen, we achieve a consistent increase in the mean overall accuracy by introducing the EOC state. Detailed inspection of the application specific results shows that there is a particularly strong increase of the mean recall for *dns* and *ssl*. As an example, the mean recall for *dns* increases from 79.3% to 93.5% for boundary set $B_1$ and 4 data packets. At the same time, the mean recall for *ssl* is improved by 14.4 percentage points reaching 81.7%. An improvement of about 2 percentage points is also observed for *ftp* whenever the EOC state is included. Regarding the other applications, the effect on the recall value is in the range of $\pm 1$ percentage points in most of the cases. Due to the better overall accuracy, we concentrate on models including the EOC state in the following discussion.

The models based on the smallest boundary set $B_3$ exhibit the worst performance. We conclude that the boundary set should have a minimum size of 3, corresponding to four payload length intervals. Moreover, the parameterizations with interval boundary sets $B_2$, $B_4$ and $B_5$ yield higher overall accuracy compared to the boundary set $B_1$. In particular, the models based on these boundary sets exhibit better performance in distinguishing the protocols *smtp* and *pop*. Based on the boundary set $B_5$ and 4 data packets, we achieve a mean recall of 87.1% and 97.8% for *pop* and *smtp*, respectively. In comparison, the model based on $B_1$ reaches 71.9% and 67.1%, only.

The models based on the interval set $B_5$ exhibit a slightly higher accuracy than models based on $B_2$ and $B_4$. On the other hand, the number of states in the models can be reduced by means of the smaller boundary set $B_4$. Depending on the scenario, this may compensate for the disadvantages in terms of accuracy. Concerning the number of considered data packets, there is a consistent increase in accuracy from 3 to 5 packets for all parameterizations considered. With 6 packets, the accuracy slightly decreases for $B_4$ and $B_5$ wheres a small increase is observed for the other settings. Therefore, we conclude that the best number of data packets depends on the interval boundaries. For the boundary sets reaching relatively higher overall accuracies, i.e. $B_2$, $B_4$ and $B_5$, the strongest increase occurs when considering 4 instead of 3 data packets. Only minor improvements can be achieved by integrating more data packets into these models, which is consistent with our earlier findings [7].

## 3.4 Classification of Test Data

In the last subsection, we obtained the best overall accuracy with Markov models considering five data packets, the new EOC state, and five payload length intervals with boundaries $B_5 = \{40, 190, 380, 620\}$. With these settings, we train a new traffic classifier using the entire set of training data (i.e. all ten folds). Thereafter, we apply the classifier to the TCP connections in the test data which represent about 2% of the original traffic. As described in Section 3.1, none of the connections in the test data is included in the training data. Connections belonging to one of the twelve applications considered in the Markov models are classified with an overall accuracy of 94.51%, which is 1.8 percentage points smaller than the average accuracy achieved in the cross-validation. The reason is the different traffic mix which now corresponds to the percentages shown in Table 1.

Table 3 shows the recall and precision values of the twelve applications as well as the proportion of unclassified connections, which is negligible in all cases. In addition, the last column indicates the labels assigned to the incorrectly classified connections. All applications are detected with a recall value equal or larger than 92%. The low precision values for *bittorrent*, *dns*, and *windowsmedia* go back to several *http* connections being classified as one of these applications. Several *smtp* connections are classified as *pop*, which explains the low precision value of *pop*.

For comparison, we train and apply another traffic classifier which considers the same number of data packets and the same payload length intervals but does not include the EOC state in the Markov models. Table 4 shows the results based on the same training and test data as used before. As can be seen, the recall value of *dns* significantly decreases from 92% with EOC state to 82% without EOC state. The recall values of *http* and *ftp* decline by about 2.4 percentage points, the one of *ssl* is reduced by 1.1 points. For the other applications, the recall is identical or only slightly decreased. The overall accuracy now is 92.70%, which is about 1.8 percentage points smaller than before. As expected, the influence of the EOC state on the application specific recall values is very similar to what we observed in the cross-validation results for this setting.

**Table 3** Classification results of test data for $B_5 = \{40, 190, 380, 620\}$ with EOC state

| | Recall | Precision | Unclassified | Most frequently misclassified as |
|---|---|---|---|---|
| bittorrent | 0.9428 | 0.0906 | 0.0000 | http, ssl |
| dns | 0.9200 | 0.1776 | 0.0000 | windowsmedia, bittorrent, ssl |
| flash | 0.9777 | 0.9535 | 0.0000 | http, ssl |
| ftp | 0.9840 | 0.8145 | 0.0000 | http |
| http | 0.9428 | 0.9957 | 0.0008 | windowsmedia, ssl, bittorrent |
| imap | 0.9431 | 0.9880 | 0.0000 | smtp, pop, ssh |
| pop | 0.9729 | 0.4298 | 0.0000 | smtp |
| rdp | 1.0000 | 0.9130 | 0.0000 | |
| smtp | 0.9354 | 0.9975 | 0.0003 | pop |
| ssh | 0.9909 | 0.9977 | 0.0000 | pop |
| ssl | 0.9406 | 0.8954 | 0.0000 | bittorrent, http, windowsmedia |
| windowsmed | 1.0000 | 0.1923 | 0.0000 | |

**Table 4** Classification results of test data for $B_5 = \{40, 190, 380, 620\}$ without EOC state

| | Recall | Precision | Unclassified | Most frequently misclassified as |
|---|---|---|---|---|
| bittorrent | 0.9428 | 0.0518 | 0.0000 | ssl |
| dns | 0.8200 | 0.1872 | 0.0000 | ssl, windowsmedia, bittorrent |
| flash | 0.9788 | 0.8853 | 0.0000 | http, ssl |
| ftp | 0.9600 | 0.9448 | 0.0000 | http |
| http | 0.9193 | 0.9948 | 0.0008 | ssl, bittorrent, windowsmedia |
| imap | 0.9431 | 0.9880 | 0.0000 | smtp, pop, ssh |
| pop | 0.9729 | 0.4298 | 0.0000 | smtp |
| rdp | 1.0000 | 0.9130 | 0.0000 | |
| smtp | 0.9357 | 0.9975 | 0.0000 | pop |
| ssh | 0.9909 | 0.9977 | 0.0000 | pop |
| ssl | 0.9295 | 0.8435 | 0.0000 | bittorrent, http, windowsmedia |
| windowsmed | 1.0000 | 0.1923 | 0.0000 | |

In the following, we continue our discussion of the classification results obtained with EOC state. As described in Section 3.1, OpenDPI provided more than one label for several TCP connections. In most of these cases, the first label was *http*, followed by a label related to multimedia content, such as *flash* or *mpeg*. Although we did not include any of these connections in the training data, more than 98% of those contained in the test data are appropriately classified as *http*.

Now, we have a closer look at the traffic which was marked as *unknown* by OpenDPI. For these connections, we can use the server port number as a weak indicator of the application. As already mentioned, a large proportion of *unknown* connections is directed to port 10050 on one host. All of these connections are classified as *rdp*, which is Microsoft's remote desktop protocol. Almost all traffic to port 2703 (Razor) is classified as *pop*. In both cases, there seems to be a strong similarity to the corresponding protocols although the signatures of OpenDPI do not match.

Regarding *unknown* connections to well-known port numbers, the classification results are often as expected. For example, most of the connections to port 143 are classified

as *imap*. About half of the connections to ports 25 and 110 are recognized as *smtp* and *pop*, respectively. 59% of the connections to port 443 (intended for https) and 100% of the connections to port 1352 (Lotus Notes) are classified as *ssl*. Although Lotus Notes is likely to use SSL, OpenDPI did not label any of these connections as *ssl*. 3.3% of the connections labelled *unknown* go to port 80. Interestingly, only 13% of them are classified as *http* while 32% are considered as *ssl* and 12% remain unclassified. A probable explanation for not finding any conformance with HTTP is that port 80 is used by other protocols and applications.

## 4 Conclusion

In this paper, we presented and discussed a TCP traffic classification approach using observable Markov models. Departing from our previous work [7], we proposed the integration of an additional state which further improves the classification accuracy. We tested and evaluated the improved classifier under real-world conditions using TCP traffic captured in the Munich Scientific Network. By means of cross-validation, we first evaluated various parameterizations and proposed methods to systematically derive the interval boundaries needed for quantizing payload lengths. Thereafter, we demonstrated that our approach is able to classify a representative traffic mix into twelve application classes with very high accuracy.

Our traffic classifier is based on Markov models which are intuitive and easy to understand. As another advantage, the training and classification complexity is low. A time-consuming task is the greedy search for appropriate payload length intervals. Future work will show whether this task actually needs to be repeated for training data from different networks or whether the traffic characteristics across networks are consistent enough to yield good classification results using the same payload length intervals or even the same Markov models.

Regarding the design of our classifier, we still see room for improvements, in particular regarding the mapping of payload length to states. We currently use constant payload length intervals for all packet positions. Instead, the best set of interval boundaries could be determined separately for each packet position, which may lead to better classification results. As a downside, this flexibility increases the complexity of finding appropriate values for all boundaries.

Most kinds of unencrypted traffic can be correctly classified based on port numbers or signatures. Nevertheless, it is useful to deploy our traffic classifier in parallel in order to determine the most likely application for those connections which remain unclassified by DPI. Apart from that, we are currently investigating the classification of encrypted traffic with the goal to distinguish different applications running on top of SSL and TLS. For these kinds of traffic, our Markov-based classifier possibly provides useful information while DPI is only able to detect SSL or TLS records.

## Acknowledgments

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

## References


1 Hui Dai, Gerhard Münz, Lothar Braun, and Georg Carle. TCP-Verkehrsklassifizierung mit Markov-Modellen. In *Leistungs-, Zuverlässigkeits- und Verlässlichkeitsbewertung von Kommunikationsnetzen und Verteilten Systemen, 5. GI/ITG-Workshop MMBnet 2009*, Hamburg, Germany, September 2009.

2 Alberto Dainotti, Walter de Donato, Antonio Pescapè, and Pierluigi Salvo Rossi. Classification of network traffic via packet-level hidden markov models. In *Proc. of IEEE Global Telecommunications Conference (GLOBECOM) 2008*, New Orleans, LA, USA, November 2008.

3 Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley & Sons, 2001.

4 Alice Este, Francesco Gringoli, and Luca Salgarelli. On the stability of the information carried by traffic flow features at the packet level. *Computer Communication Review*, 39(3):13–18, 2009.

5 Juan M. Estevez-Tapiador, Pedro Garcia-Teodoro, and Jesus E. Diaz-Verdejo. Stochastic protocol modeling for anomaly based network intrusion detection. In *Proc. of IEEE International Workshop on Information Assurance (IWIA) 2003*, Darmstadt, Germany, March 2003.

6 Hyun-chul Kim, Kimberly Claffy, Marina Fomenkov, Dhiman Barman, Michalis Faloutsos, and KiYoung Lee. Internet traffic classification demystified: Myths, caveats, and the best practices. In *Proc. of ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT) 2008*, Madrid, Spain, December 2008.

7 Gerhard Münz, Hui Dai, Lothar Braun, and Georg Carle. TCP traffic classification using Markov models. In *Proc. of Traffic Monitoring and Analysis Workshop (TMA) 2010*, Zurich, Switzerland, April 2010.

8 Thuy T. T. Nguyen and Grenville Armitage. A survey of techniques for internet traffic classification using machine learning. *IEEE Communications Surveys & Tutorials*, 10(4):56–76, 2008.

9 OpenDPI Homepage. http://www.opendpi.org/, 2010.

10 Vern Paxson. Bro: A System for Detecting Network Intruders in Real-Time. *Computer Networks*, 31(23-24):2435–2463, December 1999.

11 Vipul's Razor Homepage. http://razor.sourceforge.net/, 2010.

12 Charles Wright, Fabian Monrose, and Gerald Masson. HMM profiles for network traffic classification (extended abstract). In *Proc. of Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC) 2004*, pages 9–15, Fairfax, VA, USA, October 2004.