Multimedia Tools and Applications (2019) 78:18271–18291
https://doi.org/10.1007/s11042-019-7183-6

Identification of adaptive video streams based on traffic correlation

Arkadiusz Biernacki[1]

Received: 27 April 2018 / Revised: 15 December 2018 / Accepted: 6 January 2019 /
Published online: 22 January 2019

Abstract
Due to the popularity of Dynamic Adaptive Streaming Over HTTP (DASH), broadband and Internet service providers' links transmit mainly multimedia content. As the most popular providers encrypt their video services, the attempts to identify their traffic through Deep Packet Inspection (DPI) encounter difficulties. Therefore, encrypted DASH traffic requires new classification methods. In this work, we propose to identify DASH traffic taking into account statistical dependencies among video flows. For this purpose, we employ cluster analysis which can identify groups of traffic flows that show similarity using only the application level information. In our work, we applied three unsupervised clustering algorithms, namely MinMax K-Means, OPTICS and AutoClass, to classify video traces obtained from an emulated environment. The experimental results show that the employed algorithms are able to effectively distinguish video flows generated by different play-out strategies. The classification performance depends on the network conditions and parameters of the learning process.

Keywords Network traffic · Adaptive video · Multimedia communication · Video streaming

1 Introduction

The growing popularity of DASH service floods broadband and Internet service providers' links with multimedia content. Identification and categorization of this type of traffic is an important part of network management tasks which include flow prioritization, traffic shaping, policing, and diagnostic monitoring. Similar to network management tasks, many network engineering problems such as workload characterization and modelling, capacity planning, and route provisioning also benefit from accurate identification of network traffic.

✉ Arkadiusz Biernacki
arkadiusz.biernacki@polsl.pl

[1] Institute of Computer Science, Silesian University of Technology, Akademicka 16, 44-100 Gliwice, Poland

For example, a network operator may want to identify and throttle DASH services to manage its bandwidth budget and to ensure good performance of business-critical applications.

The classical approach to traffic classification relies on mapping applications to well-known port numbers and has been very successful in the past. However, DASH uses popular 80 and 443 TCP ports and its data is multiplexed with other HTTP based traffic, thus, port-based identification for these services are not appropriate. Another popular approach relies on DPI, however its effectiveness for encrypted traffic is limited [13, 45].

DASH players implement stream-switching (or multi-bit-rate): the content, which is stored on a web server, is encoded at different bit-rate levels, then an adaptation algorithm selects the video level, which is to be streamed. The algorithm takes into account a state of a video player, for example, a level of player's buffer, or a state of the network environment, for example, amount of available bandwidth [41]. In the current approaches, usually the video player alone chooses suitable video quality and is responsible for an adaptation to the network environment. This allows the player to independently select its playback quality without the support of any additional control protocols and communication with a server.

However, the above client-driven approach has some drawbacks. As each player strives to optimise its individual quality, they implement competing play-out algorithms which try to outsmart one another. Thus, when data streams, which are downloaded simultaneously by several players, traverse the same path in the network, the players compete for the available bandwidth. Such scenario quite often takes place when there are several concurrent sessions initiated by video players located within an Internet Service Provider (ISP) network, as presented in Fig. 1. As it was shown in [9], when the same play-out strategies are shared among the video players, the traffic generated by these strategies tends to be positively correlated what leads to an increase in variability of whole aggregated traffic transmitted through a given network path. This correlation may be used to classify different play-out strategies what is the main contribution of our work.

In order to classify DASH traffic, we employed three clustering algorithms: MinMax K-Means [7], OPTICS [4] and popular AutoClass. Although, these algorithms use an unsupervised learning mechanism, each of them is based on different clustering principles. The unsupervised learning aims at discovering unknown relationships and patterns in video streams. Compared to the supervised learning, this approach does not require labelled training data.

We conduct the performance study using an emulation model what allows us to methodologically explore the behaviour of the examined system over a wide range of parameter settings, which would be a challenging task to conduct such experiments only on a

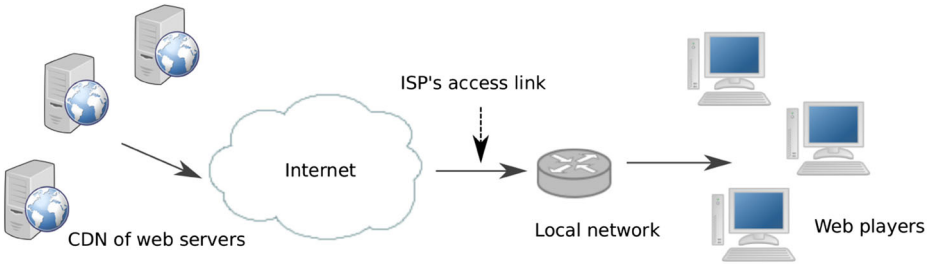

Fig. 1 HTTP streaming scenario. Video traffic is multiplexed on the ISP access link

real-network. Simultaneously, as the emulation is performed in a laboratory environment, we are able to preserve much of the network realism because we conduct experiments using real hardware and software, what permits us to maintain a decent level of accuracy for the obtained results. The players implement four different play-out algorithms which mimic the behaviour of real commercial video players.

We compare the performance of traffic classification manipulating with parameters of clustering algorithms and changing the configuration of the network environment. As we will show, the proposed approach achieves decent accuracy and precision and may be used as a supporting method for other hybrid classification techniques.

2 Application level flow control

In most modern video systems based on HTTP, the video file is divided into chunks of fixed length and the server pushes them sequentially to the client with a frequency which allows obtaining a data transmission rate little higher than the video-bit rate of the transmitted content. As a result, the transmitted traffic creates an ON-OFF pattern, where the ON and OFF periods may have a variable length [1]. The extension of this idea is an adaptive streaming, which offers more flexibility when a network environment is less stable, for instance in wireless mobile networks. With this approach, it is possible to switch the media bit rate (and hence the quality) after each chunk is downloaded and adapt it to the current network conditions [42]. In this approach, a video stream is also divided into segments, but this time, they are encoded in multiple quality levels, called representations. Based on an estimation of the available throughput, the client may request subsequent segments at different quality levels which depend on network conditions between the client and server, as drafted in Fig. 2. The algorithm deciding which segment should be requested in order to optimize the viewing experience is the main component and a major challenge in adaptive streaming systems because the client has to properly estimate, and sometimes even predict, network conditions, for example, the dynamics of the available throughput. Furthermore, the client has also to control a filling level of its local buffer in order to avoid underflows resulting in playback interruptions.

The commercial video systems usually do not share many details about the employed algorithms, thus, in our experiments, we use the template of an adaptive streaming algorithm based on a bandwidth estimation which is a part of the open-source software described in [36]. Using this template, we implemented four open-source adaptive algorithms which are employed in popular software [22] or, at least, gained a fair amount of publicity [28, 33, 35].

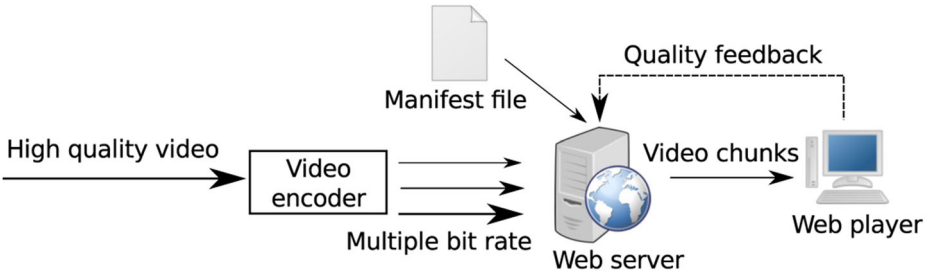

Fig. 2 Architecture of a video adaptive system based on HTTP

2.1 Play-out algorithms

The first implementation uses the Microsoft Smooth Streaming (*MMS*) algorithm, which is based on the open source version of the algorithm of the MSS video player and is extensively described in [22]. The algorithm evaluates the status of the play-out buffer and its decision based on the results of measurement of network throughput and comparison of the buffer state. The buffer is divided into three thresholds denoted by the authors as: a lower, an upper threshold and a panic threshold.

The second approach, *Festive* [28], similarly to the mentioned *MSS* approach, takes into account an estimation of network capacity on which future bit-rate update decisions can be made. The algorithm also uses a gradual switching strategy; i.e., each switch is only to the next higher or lower level. Unlike *MMS*, *Festive* uses randomised scheduling and takes into account also fairness among video players.

The third heuristic relies on the *PANDA* (Probe AND Adapt) algorithm proposed in [33]. The algorithm takes TCP download throughput as an input to the adaptive algorithm only if the measurement is an accurate indicator of the fair-share bandwidth. The video quality is not directly related to network throughput but to its average, which in turn determines the selected video bit-rate and the request time between video chunks.

The last proposition employs so-called *LOLYPOP* (Adaptation Algorithm for Low-Delay Live Streaming) [35]. The aim of *LOLYPOP* is to operate with a transport latency of a few seconds. To reach this goal, *LOLYPOP* tries to predict TCP throughput on multiple time scales: from 1 to 10 seconds. In addition to the imposed latency constraint, the algorithm heuristically improves the quality of experience by maximising the average video quality as a function of the number of skipped segments and quality transitions.

The list of the algorithms with their corresponding symbols used in the further experiments is presented in Table 1.

3 Related works

The problem of network traffic classification can be perceived as a never-ending race between application developers, on the one side, and the ISPs supported by the research community, on the other.

Several techniques have been popularly employed for traffic classification over the years. The simplest approach is based on TCP port classification. This technique was to some extent quite successful because many applications used fixed port numbers assigned by Internet Assigned Numbers Authority (IANA). However, over time this approach has shown to have more and more limitations. A growing number of applications had been not attached to fixed port numbers, instead using dynamically assigned port numbers assigned by negotiation mechanisms, e.g. to hide from firewalls or network security tools. To address the

Table 1 Play-out algorithms used in the experiments

Sym.	Algorithm
A	*MMS*
B	*Festive*
C	*PANDA*
D	*LOLY*

aforementioned drawbacks of port-based classification, several DPI techniques have been proposed [11, 23]. In this approach, packet payloads are inspected to determine whether they contain characteristic signatures or patterns of known applications. As the search for patterns, based often on regular expressions, inside a data may seem to be technically not complicated, there are important limitations of this idea. Firstly, it works in a sequential way: a packet after a packet needs to be analysed what requires time and a fair amount of memory. Secondly, this idea is very difficult or even impossible when dealing with encrypted traffic which volume has been increasing. In this case, DPI is concentrated mainly on recognizing TLS/SSL handshake parameters that help recognize the application content types (video, VoIP, etc.) or the application name [13, 45].

Thus, researchers have been proposing new methods of traffic classification which have been capable of inferring application-level usage patterns without relying exclusively on the TCP ports matching or on DPI. Some of the proposed approaches classify traffic by recognising statistical patterns in externally observable attributes of the traffic, e.g. packet lengths or inter-packet arrival times. Their aim is either clustering traffic flows into groups that have similar traffic patterns, or classifying applications generating the flows. For this purpose, many works apply so-called machine learning (ML) techniques. The ML classifier is trained to associate sets of attributes with known traffic classes, thus creating rules, which are applied to classify unknown traffic. Every ML algorithm has a different approach to sorting and prioritising sets of features, which leads to different dynamic behaviours during training and classification. A comprehensive review of ML methods can be found, e.g. in [37, 38].

Traffic classification can be achieved using either supervised or unsupervised learning. In the former case, there is a need to acquire labelled training data-sets, something that can be challenging in computer networks due to the difficulty in obtaining accurately annotated network flow samples across a broad range of applications and the rate at which new applications can appear. As a result, many of this approaches are limited to coarse-grained classifications of traffic originated from, e.g. web browsing, P2P or VoIP [3, 30]. An alternative is an employment of unsupervised ML which processes unlabelled data. The goal is to find unknown relationships in the data, finding similarity patterns among several observations. Unsupervised ML is usually applied for clustering tasks, where it is required to divide the data into different groups according to similarities in their features. In particular, this form of ML learning can be applied to discover traffic from previously unknown applications [19].

There is a number of algorithms that can be used in unsupervised ML and there is abundant literature on their application for traffic classification. A popular approach based on the K-Means algorithm is presented in [7]. In [43] the authors created a methodology and a framework called *AppScanner* for automatic fingerprinting and real-time identification of Android application in encrypted network traffic. They compared Random Forest and SVC approaches because both of them were found to be particularly suited for network flow classification. The authors of [15] focused on identification of traffic generated by Google Hangout, Google Plus and Gmail. Three algorithms were compared: Naive Bayes, AdaBoost and j48 decision tree.

However, the clustering methods suffer from a problem of mapping from a large number of clusters to real applications. This problem is very difficult to solve without knowing any information about real applications. Therefore, in many cases, a combination of unsupervised and supervised methods, refereed as semi-supervised learning, can also be used with both labelled and unlabelled data [47]. It can work with a data-set where the majority of the instances are unlabelled if a small number of labelled data instances exists that permits the

mapping from the identified clusters in the overall data-set into the different classes, trying to overcome the difficulties in obtaining labelled data.

It was shown in [2] that statistical identification approaches deployed for unencrypted traffic are not always valid for the encrypted cases as the new encryption methods and protocols differ in behaviour from one application to another. Thus, encrypted traffic, which is growing rapidly in the Internet, needs often separate approach. In [45] the authors present a broad comparison of different approaches for encrypted traffic classification. The K-Means approach proved to have good average accuracy for quality classification of encrypted adaptive video in [17]. In [18] the above problem with video quality classification was approached using ML methods. The dataset in these works was obtained with different bit rates, in order to examine the classifier performance in various circumstances. K-Means is also used in [16] for P2P traffic identification. Several comparisons among unsupervised algorithm applied to encrypted traffic classification can be found also in [6, 40].

The above literature shows that there is a broad spectrum of techniques for classification of encrypted network traffic. Hence, the amount of attributes which can be taken into account is relatively limited. The main contribution of our work is the employment of correlation among flows as an attribute which can support the identification of adaptive video streams. Thus, in our work, we discover correlation in the traffic flows and we apply it into the classification process. Contrary to conventional classification methods, which treat the traffic flows as the individual and independent instances, we demonstrate that the correlation information can significantly improve the classification performance, especially when the network conditions are variable.

The correlation-based classification is popularly used in some domains, especially in physics [31] or econometrics [39]. When it comes to network traffic this technique was used in [48] for classification of general Internet traffic. The closest work is [9] where the author tried to reduce correlation among video flows in order to increase the performance of video system.

In our work, we use for the classification three tested and evaluated algorithms, namely MinMax K-Means, OPTICS and AutoClass, applied in a number of works. The usage of well-know clustering algorithms increases the chances of reproducibility of our experiments. Furthermore, the produced patterns more likely reflect the internal structure and properties of gathered data independently of a clustering approach than are a result of tweaking and optimisation of a complex algorithm. Simultaneously, these algorithms were not broadly used for network traffic clustering and no performance comparison among them has been made in this field.

4 Traffic clustering

The cluster analysis is a popular method for identifying classes among a group of objects and has been used as a tool in many fields such as biology, finance, and computer science. In contrast to the classification techniques using pre-defined classes of training objects, clustering methods are not supported by this information. Instead, they discover natural patterns in the data using internalised heuristics. Clustering splits objects into groups taking into account distance among the objects calculated by a specific metric, e.g. the Euclidean metric.

The clustering of video quality traces is a special type of multidimensional time series clustering. A network trace is essentially classified as time series because its feature

values change as a function of time, which means that the value of each point of time series is a chronologically made observation. Video quality traces generated by many connections are a type of temporal data which is naturally high dimensional and large in data size. Formally, given a dataset of n video quality traces data $\mathbf{Q} = \{Q_1, Q_2, \ldots, Q_n\}$, the process of unsupervised partitioning of $\mathbf{Q}$ into $\mathbf{C} = \{C_1, C_2, \ldots, C_k\}$ in such a way that homogeneous quality traces are grouped together based on a certain similarity measure, is called the clustering of the quality traces. Then, C_i is called a cluster, where $\mathbf{Q} = \bigcup_{i=1}^{k} C_i$ and $Q_i \cap Q_j = \emptyset$ for $i \neq j$.

The groups identified in such manner may be exclusive or they may be overlapping. They may also be hierarchical, where there is a division of instances into groups at the top level, and then each of these groups is refined further. In our work, we use two popular clustering methods: hierarchical and centroid-based clustering.

The hierarchical clustering generates a hierarchical grouping of instances. Using an agglomerative algorithm, which considers each item as a cluster, one incrementally merges the clusters using so-called bottom-up approach producing a dendrogram. The algorithm begins by assigning each object to its own cluster, then, at each step, the two clusters that are the most similar are merged. The algorithm continues until all of the clusters have been merged. The evaluation of the generated division is subjective, one simply assess the quality of clusters inspecting the generated dendrogram. However, if the data labels are known in advance, it is also possible to make objective statements of the clustering quality.

The centroid-based algorithms form clusters in numeric domains, partitioning objects into disjoint clusters. The centroid of a cluster can be thought of as the pure type the cluster represents, whether that object exists in the reality or is just an abstract construct. The centroid may be a particular data point in the cluster or may be a point in the convex hull of the cluster, such as the cluster mean. Given an initial data division, centroid-based methods find the centroids of the clusters, reassign the objects to new clusters defined by the distance to the centroids, and then repeat the procedure. The similarity between two clusters is the similarity between their respective centroids. Thus, aside from an initial clustering, a clustering algorithm has to define what centroids and what distance among them are to be used, and how many iterations to execute.

From a number of the centroid-based algorithms we used in our work we used MinMax K-Means, OPTICS and AutoClass.

4.1 MinMax K-Means

The MinMax K-Means algorithm is based, as the name suggest, on the K-Means algorithm which belong to partition based algorithms. In partition based algorithms, the single data object is explicitly assigned to one and only one cluster. The main idea behind K-Means is to define k centroids, one for each cluster. These centroids should be placed in a deliberate way because the final result is influenced by the location. The procedure starts at those k centroids, and each of them absorbs nearby points, based on a defined distance. Thus each point belonging to a given data set is associated with its nearest centroid. When no point is pending, new centres of clusters are re-calculated taking into account the absorbed points. After obtaining k new centroids, a new binding has to be done between the same set of points and the nearest new centroid. Then, the procedure is being repeated: new centers are allowed to absorb nearby points based on a defined distance and the coordinates of the new centres are re-calculated. As a result of this loop, k centroids shift to new locations step by step until no more significant changes are noticed. Finally, this algorithm aims at

minimizing the objective function, in this case, a squared error function which has the form of

$$L_a(C) = \arg\min_{C} \sum_{i=1}^{k} \sum_{\mathbf{Q} \in C_i} ||\mathbf{Q} - \mu_i||^2, \tag{1}$$

where $\mathbf{Q} = \{Q_1, Q_2, \ldots, Q_n\}$ is a dataset of n video quality traces and μ_i are the coordinates of the centroid of the cluster C_i. However, the K-Means algorithm has some disadvantages, among them sensitiveness to the selection of the initial centroids. An inappropriate initialization may lead to a slower convergence and misclassified clusters. The MinMax K-Means approach tries to make the algorithm's result less dependent on the initial centroids selection. The algorithm starts also with random initial centers but instead of minimizing the sum of internal variance of clusters like in (1), it attempts to minimize the maximum internal variance of clusters

$$L_a(C) = \arg\min_{C} \max_{1 \le i \le k} \sum_{\mathbf{Q} \in C_i} ||\mathbf{Q} - \mu_i||^2. \tag{2}$$

Each cluster is weighted so that higher weights are assigned to the cluster with larger internal variance. By applying this method, the results become less dependent on the initialization and, thus, the quality of clustering increases even when the initial centers of clusters are not selected optimally [44]. This approach can be justified as following: the summation over all clusters in (1) allows to achieve similar sum values either by having a few clusters with a large variance that are counterbalanced by the others clusters with small variance, or by having a moderate variance for all clusters. However in (2), having a few clusters with large variance leads to a higher objective value. Hence in (2), clusters with a large variance are avoided and the solution space is restricted towards clusters that have similar variances.

4.2 OPTICS

Ordering points to identify the clustering structure (OPTICS) represents the family of density-based algorithms. These algorithms consider clusters as dense areas of objects that are separated by less dense areas. Density-based algorithms have the ability to find clusters of arbitrary shapes, what is an advantage over partition-based algorithms (e.g. K-Means, MinMax K-Means) which are limited to recognize only spherical shaped clusters.

The basic idea of OPTICS is similar to DBSCAN [21] but it addresses one of DBSCAN's significant weaknesses, i.e. the ability to detect meaningful clusters in data which have variable density. In order to handle this problem, the algorithm uses two concepts: a core-distance and reachability-distance. These concepts depend on two input parameters: ϵ which is the distance around an object that defines its ϵ-neighbourhood, and *minPts* which is a minimum number of points constituting ϵ-neighbourhood.

For a given object p, when the number of objects within the ϵ-neighbourhood $N_\epsilon(p)$ is at least *minPts*, then p is defined as a core object. The core-distance of p is defined as the smallest distance ϵ' between p and an object in its ϵ-neighborhood such that p would be a core object with respect to ϵ' if this neighbour is contained in $N_\epsilon(p)$. If there not enough objects in $N_\epsilon(p)$, i.e. $|N_\epsilon(p)| < minPts$, the core distance is undefined. This is illustrated in Fig. 3.

The reachability-distance of an another object o from the object p is either the distance between o and p, or the core distance of p, whichever is bigger assuming that $|N_\epsilon(p)| \geq MinPts$. If the latter condition is not fulfilled, the core distance is undefined. Thus, both

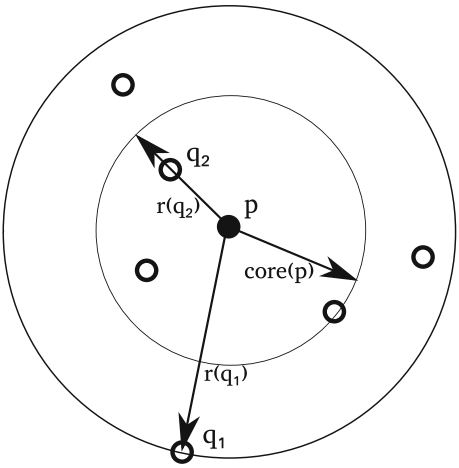

Fig. 3 The concepts of a core-distance *core(p)* and reachability-distance for $r(q_1)$ and $r(q_2)$ for *minPts* set to 4 (including p)

the core-distance and the reachability-distance are undefined if no sufficiently dense cluster (w.r.t. ϵ) is available.

As a result, the OPTICS algorithm produces a list of objects in a particular ordering, annotated with their smallest reachability distance. Since grouped objects have a low reachability distance to their nearest neighbour, using a special kind of dendrogram, so called reachability-plot, it is possible to identify clusters in the data.

4.3 AutoClass

AutoClass is a clustering algorithm based on probabilistic model which assigns instances to classes probabilistically, not deterministically as in the case of MinMax K-Means or OPTICS. To build the probabilistic model, the clustering algorithm determines the number of clusters and the parameters that govern the distinct probability distributions of each cluster. To accomplish this, AutoClass uses the Expectation Maximization (EM) algorithm. To find the global maximum, AutoClass repeats EM searches starting from pseudo-random points in the parameter space. The model with the parameter set having the highest probability is considered to be the best. AutoClass uses a Bayesian score to determine the best set of parameters to use for the probabilistic model and allows for an automatic selection of the number of clusters. The Bayesian score is based on intra-cluster similarity and inter-cluster dissimilarity. Also, the Bayesian score penalizes models with more clusters to minimize potential over-fitting.

4.4 Measures of similarity

In order to cluster network traces, one needs to define notions of similarity. Similarity can be thought of as distance so that the points in a cluster are closer to each other than they are to the points in other clusters. Distance measures provide quantification for the similarity between two time series. Calculating distances, as well as cross-distance matrices, between time-series objects is one of the cornerstones of any time-series clustering algorithm. The algorithm computes distance between all pairs of data objects applying a suitable distance function. In our work, the distance will be based on the cross-correlation among network traces.

Thus, in the first step of our analysis, we are interested in a correlation of traffic generated by particular video strategies. While adjusting video bit-rate to a current network environment, each of the players generates time series representing respective quality levels q_i which correspond to video bit-rates. From the above set of time series, we compute the correlation coefficient between any pair of players as

$$\rho_{ij} = \frac{\langle Q_i Q_j \rangle - \langle Q_i \rangle \langle Q_j \rangle}{\sqrt{(\langle Q_i^2 \rangle - \langle Q_i \rangle^2)(\langle Q_j^2 \rangle - \langle Q_j \rangle^2)}}, \tag{3}$$

where $Q_i, Q_j \in \{Q_{min}, \ldots, Q_{max}\}$ represent a stochastic process whose elements are quality levels of video q_i, q_j received by players i and j respectively, and $\langle Q_i \rangle$ is an average of the vector Q_i. By definition, ρ_{ij} can vary from -1, what denotes completely negative correlated (or anti-correlated) video bit-rates, to 1, what denotes completely positive correlated video quality levels. When $\rho_{ij} = 0$, the quality of video streamed to i-th and j-th players are uncorrelated. The correlation coefficient ρ_{ij} (3) is computed between all the possible pairs of players downloading video from the emulated CDN. As a result, we obtain a symmetrical matrix characterised completely by $n(n-1)/2$ correlation coefficients with $\rho_{ii} = 1$ in the main diagonal.

However, we cannot input the correlation matrix directly to the clustering algorithms because the matrix elements in their original form are not suitable as a distance measure, for example, a distance between the same two elements would have been one ($\rho_{ii} = 1$) instead of zero. Generally, the elements of the correlation matrix do not fulfil the three axioms that define a metric: minimality, symmetry and triangle inequality. Hence, the matrix must be transformed in order to define a genuine metric. Following [34] we use the transformation

$$d_{ij} = \sqrt{2(1 - \rho_{ij})}. \tag{4}$$

With this choice d_{ij} fulfils the three axioms of a metric distance: a) $d_{ij} = 0$ if and only if $i = j$; b) $d_{ij} = d_{ji}$ and c) $d_{ij} \leq d_{ik} + d_{kj}$. The draft of the proof is presented in [8] in Appendix I.

4.5 Evaluation

One of the advantages of unsupervised clustering is the automatic identification of classes. Nevertheless, the obtained cluster should be labelled in order to be correctly mapped to applications.

To assess the clustering results, we adopt the majority heuristic to label clusters based on ground truth. That is, we label clusters with the most popular play-out strategies presented in them. For this purpose, we adopted the labeling schema used in [37]. Let $C = \{C_1, \ldots, C_k\}$ be the clusters and $S = \{S_1, \ldots, S_m\}$ be the actual play-out strategies. The labeling function $L_b : C \rightarrow S$ associates a class label to each cluster

$$L_b(C) = \arg\max_{S_j \in S} \sum_{\mathbf{x_i} \in C} \Theta(\beta(\mathbf{x_i}), S_j), \tag{5}$$

where $\beta(\mathbf{x})$ returns the actual play-out strategy of the given flow sample $\mathbf{x}$ and $\Theta(X, Y)$ is defined as

$$\Theta(X, Y) = \begin{cases} 1 & \text{if } X = Y \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

To estimate the performance of the clustering algorithms we employed the accuracy measure which describes how well the clustering algorithm is able to create clusters that contain

only a single video traffic category. The traffic class which instances inhabit the majority of objects within a cluster labels this particular cluster. Assigning two similar traces to the same cluster is a true positive (TP) decision while assigning two different strategies to different clusters is a true negative (TN) decision. During this process, it is also possible to assigns two dissimilar strategies to the same cluster what is considered a false positive (FP) decision, or to assign the same two strategies to different clusters what is a false negative (FN) decision. These objects which have not been assigned to the cluster are labelled as noise. Consequently, accuracy is the fraction of correctly classified objects and is computed as follows:

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}. \tag{7}$$

When there are problems with the proper classification of the object, the accuracy goes towards zero. When the opposite situation takes place, the measure heads to one.

One of the drawbacks of (7) is that a system tuned to maximize accuracy can appear to perform well by simply deeming all objects non-relevant (FP and TN) to all queries. Therefore, the second applied complimentary measure is precision. The precision concentrates on the evaluation of true positives, asking what percentage of relevant objects have been identified. Thus, this metric indicates the fraction of instances from a particular category that are correctly classified from the total amount of instances classified as that category. The precision is computed as follows

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{8}$$

5 Laboratory set-up

In order to acquire traffic traces, which were generated by the algorithms mentioned in Section 2.1, we prepared a test environment emulating content distribution network (CDN), which in the real world is applied to deliver video to multiple users. The environment consists of network environment emulators (NEE), web servers, video players and a measurement point located in an edge router, as shown in Fig. 4.

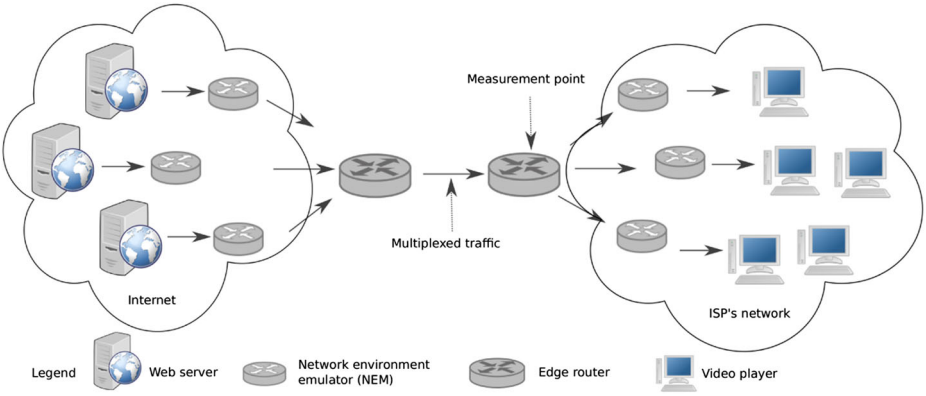

Fig. 4 An emulated CDN network which consist of network environment emulators, web servers, video players and a measurement point located in an edge router

As the NEE, we used a network emulation node based on the built-in Linux Kernel module *netem* [25]. The module allows for manipulation with network parameters such as network bandwidth or packet delays; thus, it is possible to test the data transmission in different network environments.

The role of the web server plays Apache [5], which stores the video clips as a set of chunks. Each of the three Apache servers used in the experiment has assigned an NEE which mimics a different network environment so that there are different packet delays and network bandwidth between the servers and the edge router. The delays are described by the Weibull distribution [24, 26] with an average of 0.03 s. The network bandwidth is uniformly distributed and takes its values 4 MB/s to 12 MB/s. We assume that the servers in the CDN are connected by a high-performance wired network, therefore, the packet losses are negligent. Taking into account that according to [14] about 65%-80% of current network traffic is generated by video services, we allocated a quote of 25% of the bandwidth for background traffic which was generated by the tool presented in [10].

The background traffic is labelled and is not taken into account during the identification process. As it was stated, the goal of the present study is to obtain the taxonomy of the play-out algorithms. The problem of video traffic distinction from other traffic is broadly discussed in a number of works, some of which are surveyed in, e.g. [12].

As a video player, we chose the VLC media player with the DASH plug-in [36]. Both the player and the plug-in have an open-source code, thus, it is possible to manipulate or completely change the adaptation logic without affecting the other components. As a consequence, the plug-in enables integration of a variety of adaptation logics making it attractive for performance comparison of different adaptive streaming algorithms and their parameters. As it was mentioned, we implemented here the algorithms listed in Table 1. The players were divided into three groups and each group had assigned an NEE which imposed on the groups different network properties so that the NEE of the first group emulated properties of a wired network, the NEE of the second group emulated a wireless network and the NEE of the third group mimicked a wireless mobile network.

Taking into account the proportion of players assigned to each particular network type, we consider here three scenarios. In the first scenario, we assigned 60% of video players to wired network, and the rest was equally split between wireless and mobile environments. In the second scenario, we split the players equally among the three network types and in the third scenarios, 60% of video players operated in a mobile network, while the remaining part was equally split between wired and wireless network. As our experiment will show, the network characteristics have a certain influence on clustering performance of video streams.

The players started to download the content at random times described by an exponential distribution with mean set to 30 s. Taking into the account the findings presented in [46], the content was divided into 4 s length segments and the size of the player buffer was set to 12 s. Similarly to the server-side, on the client-side packet delays have the Weibull distribution while the bandwidth is distributed uniformly. Table 2 summarizes basic parameters of the experiment.

We transmitted several video files, acquired from [32] and presented in Table 3, through the simulation environment. Using the edge router with installed capturing software, based on Tcpdump and Libcap [27], we obtained aggregated traffic traces. Next, the traces were converted into time series represented by a point process.

We believe that the above-described methodology provides an attractive middle ground between a simulation and real network experiments. To a large degree, the emulator should be able to maintain the repeatability, reconfigurability, isolation from production networks,

Table 2 Simulation parameters

Parameter	Value(s)
Number of web servers	3
Number of NEE	6
Number of video players	120
Bandwidth at the edge routes	16, 20, 24 MB/s
Bandwidth at the server side (per a server)	4 - 12 MB/s
Packet delays at the server side	0.03 s in average
Bandwidth at the client side (per a player)	0.16 - 1 MB/s
Packet delays at the client side	0.06 s in average
Packet losses at the client side	max. 1%
Video segment length	4 s
Buffer size of players	12 s

and manageability of a simulation while preserving the support for real video adaptive applications. The output from test experiments, which included 30 video players, is presented in Fig. 5, where it is possible to notice how three randomly selected video players switch their bit-rates with the corresponding total aggregated traffic generated by all players involved in the experiment.

6 Clustering performance

Visualisation is a primary method to reveal possible relationships among video streams. Unfortunately, as our data is n-dimensional, it is not possible to visualise it directly. Nevertheless, it is customary for the multidimensional data to be fairly well represented in smaller dimensions than n. A two-dimensional representation is usually a good compromise between fidelity to the original distances and the easiness of visualisation of the objects. One of the methods which may be applied for this purpose is multidimensional scaling (MDS). MDS takes data points in a high-dimensional space and compresses them down to a lower-dimensional space, representing them as a fully connected weighted graph. MDS is a multivariate statistical method for estimating the scale values along one or more continuous dimensions such that those dimensions account for proximity measures defined over pairs of objects.

Figure 6 presents the relation among streams, generated by four play-out strategies listed in Table 1, for the network environment configured as defined in Scenario 2. In order to

Table 3 Video clips used during the experiments

Name	Genre	Bitrate levels
Big Buck Bunny	animation	100 kbit/s – 320x240, 300 kbit/s – 480x360, 700 kbit/s – 854x480, 1.2 Mbit/s – 1280x720, 2.5 Mbit/s – 1920x1080
Elephants Dream	animation	
Red Bull Playstreets	sport	
The Swiss Account	sport	
Valkaama	movie	
Of Forest and Men	movie	

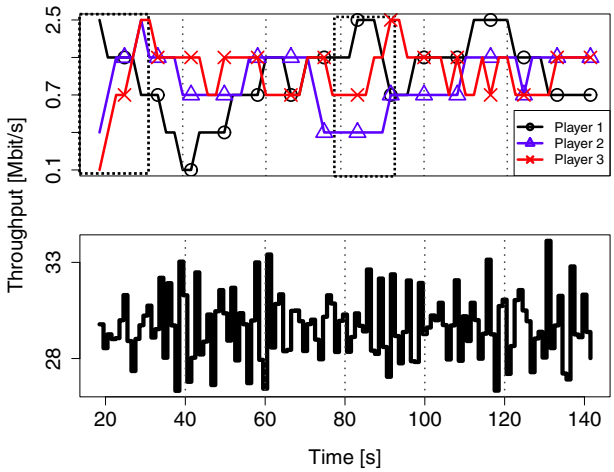

Fig. 5 Scenario with 3 randomly selected players. Total number of players = 30. Dotted areas show examples of positive correlation between Player 2 and Player 3, and negative correlation between Player 1 and the two other players

increase the presentation visibility, we sampled 80 quality traces from the total number of 120 simultaneous connections. We labelled the same play-out algorithms by the same symbols A, B, C and D; see Table 1 which lists the symbols with the corresponding play-out strategies. We may easily notice that the arrangement of the streams is not random. The streams generated by the same play-out strategies are clustered and constrained with less or more clear boundaries. Some of the clusters are easy to distinguish, e.g. the clusters labelled as 1, while the other groupings may be mixed and composed from the points representing different play-out strategies, e.g. the cluster labelled as 2.

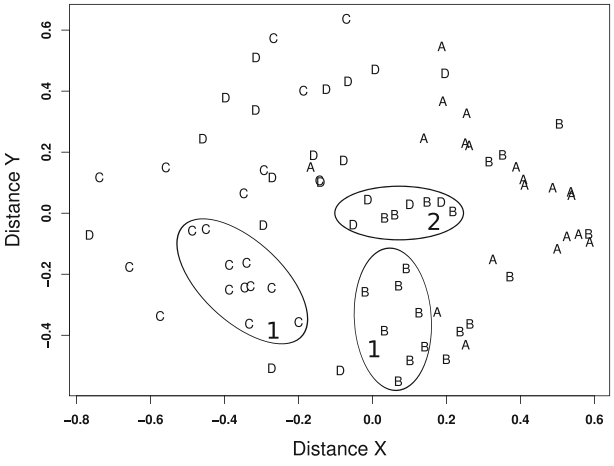

Fig. 6 Multidimensional scaling for Scenario 2. Easily distinguished clusters are labelled as 1. The grouping composed from different play-out strategies is labelled as 2

As the preliminary investigation presented in Fig. 6 indicates that there are clear relations among the video flows, we want to identify theses relation by objective measurements, applying for this purpose clustering algorithms described in Section 4.

In the next step, we used hierarchical clustering and obtained so-called dendrogram: a tree that represents the nested clusters, see Fig. 7. Similarly to Fig. 6, in order to distinguish the details, we chose a subset of the data sampling this time 40 quality traces. We can notice that the generated partition has some patterns. The dominant strategy in the first branch is A with the slight addition of strategies B and D. In the second branch, there is a mix of strategies, however, in the third branch prevails strategy C followed by strategies D and B. Strategies A and C are usually separated from each other.

Hence, the conclusion is that the most positive correlation have strategies A with B, and C with D. Also, to a lesser extent, the positive correlation can be observed between strategies A and D, and between C and B. Figure 7 shows no significant positive correlation between strategies A and C.

Hierarchical clusterings is a quite fast way to compare and contrast similarity measures since a dendrogram of size N summarizes $O(N^2)$ distance calculations [29]. Nevertheless, the hierarchical algorithms are weak in terms of quality because once divided they cannot adjust the clusters. As a result, more flexible approaches are needed. Hence, in the next analyses, we employ centroid-based solutions.

The MinMax K-Means algorithm has an input parameter k which is the number of disjoint partitions mentioned in Section 4.1. In our data sets, we would expect that there would be at least one cluster for each play-out algorithm. In addition, due to the diversity of the traffic which often does not form clearly distinguishable clusters, as presented in Figs. 6 and 7, it is reasonable to assume that the traffic will create even more disjoint partitions. Thus, the MinMax K-Means algorithm was evaluated with k initially being 4 and then being incremented of 4 in every step until k reached 20. As we have 120 data objects representing quality traces, it does not make sense to increase k further as an excessive number of clusters rises the risk of over-fitting.

The minimum, maximum, and average results for the MinMax K-Means clustering algorithm are shown in Fig. 8. Initially, when the number of clusters is small the accuracy of

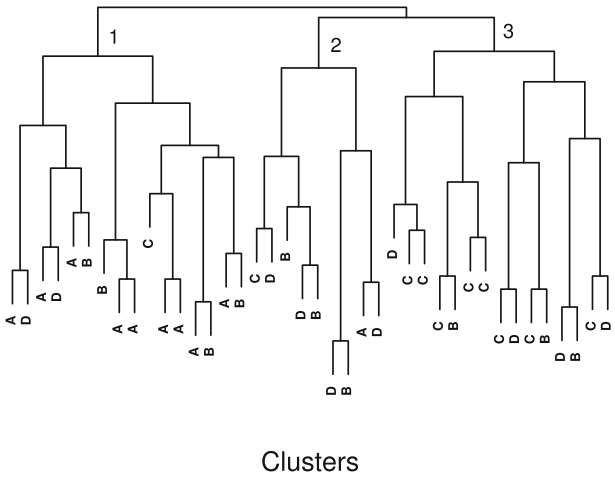

Fig. 7 Hierarchical clustering

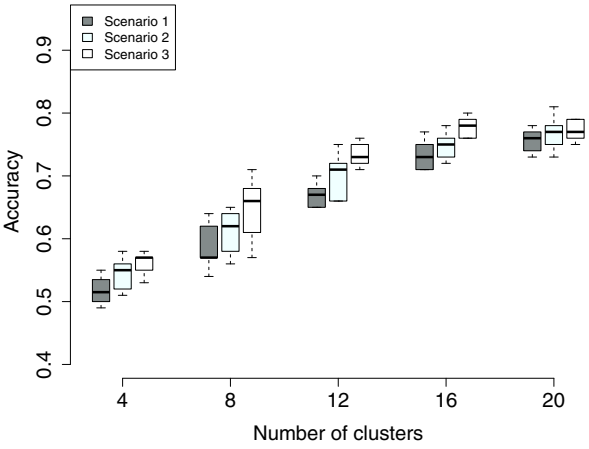

Fig. 8 Accuracy for the MinMax K-Means algorithm in a function of the number of clusters

MinMax K-Means reaches approximately 53% for the traffic generated by the wired system in Scenario 1. For the wireless and mobile networks, the accuracy increases in average about 7%-10% respectively. With the growing number of clusters, the accuracy steadily improves. However, the rate of the improvement declines after the number of clusters exceeds 16. Finally, for 20 clusters, MinMax K-Means obtains about 75%-77% depending on the network environment. The improvement in the accuracy with an increase of the clusters number may suggest that the clusters have no clear boundaries and clusters' structure may have varied density.

As described in Section 4.2, the OPTICS algorithm has two input parameters: ϵ and *minPts*. Contrary to DBSCAN, the ϵ parameter denoting the maximum size of the ϵ-neighbourhood does not play a practical role in determining the properties of the clusters. However, it is used to improve performance by OPTICS as limiting the neighborhood size

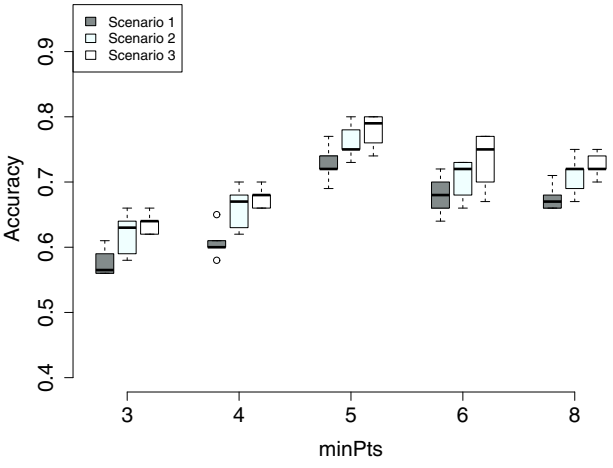

Fig. 9 Accuracy for the OPTICS algorithm in the function of the *minPts* parameter. The number of clusters is from 10 to 14

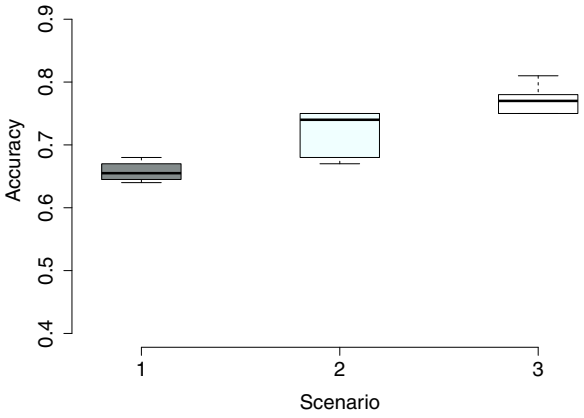

Fig. 10 Accuracy for the AutoClass algorithm

decrease amount of computations needed by the algorithm. Technically, if the parameter is not specified, the largest *minPts*-distance in the data set is used which gives the same result as infinity. Thus, we manipulate only with *minPts*, setting it between 3 and 8 which covers the range considered in [19]. The sensitivity analysis with regard to the *minPts* parameter shows that the best result is obtained for the parameter set to 5 as presented in Fig. 9. Thus, the relatively low value of this parameters suggest that there is not much noise in the examined data. The accuracy achieves about 70% for stable network conditions, and from 74% to 79% in average for more dynamic network environments. OPTICS produced in average 14 clusters for Scenario 1, 12 clusters for Scenario 2 and 10 clusters for Scenario 3.

The accuracy for the AutoClass algorithm achieves in average about 70% for video-players localised mainly in a wired network, see Fig. 10. With the migration of the players to wireless and mobile environments, the accuracy increases to about 75% and 80% respectively. As it was mentioned, for this algorithm, the number of clusters and the cluster

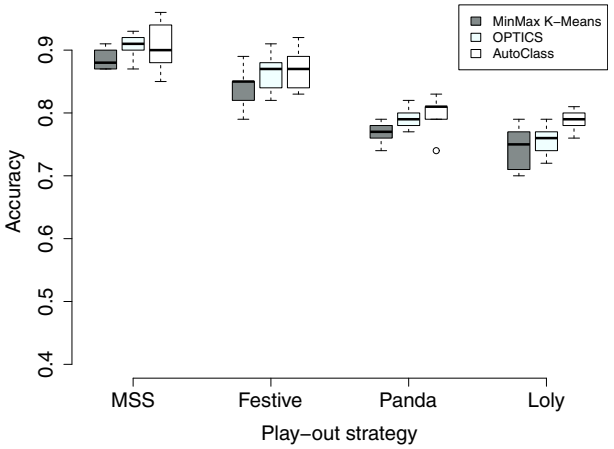

Fig. 11 Accuracy for individual classes. Parameters: MinMax K-Means: 20 clusters; OPTICS: *minPts*=5

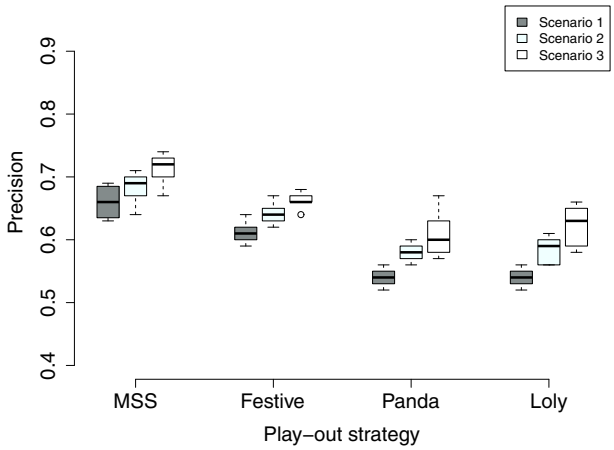

Fig. 12 Precision for different play-out strategies. Computed for the AutoClass algorithm

parameters are determined automatically. AutoClass produced in average 18 clusters for Scenario 1, 14 clusters for Scenario 2 and 12 clusters for Scenario 3.

Once the accuracy is calculated separately for every play-out strategy, the best score of about 90% achieves the *MMS* strategy, as presented in Fig. 11. A little harder to classify accurately is *Festive* which obtains in average less than 90%. Both the *PANDA* and *LOLY* strategies acquire in average less than 80%. Considering the performance of the clustering algorithms, the accuracy for *MMS* is similar for all three clustering algorithms, while for the others play-out algorithm the best accuracy gets AutoClass, followed by OPTICS and Min-Max K-Means. Nevertheless, the difference between the scores achieved by the algorithms is usually confined to several percents. In conclusion, we can observe that the algorithms with a simpler adaptation mechanism are easier to classify, even for a less advanced clustering algorithm. The more complicated adaptive mechanism generate probably less correlated traffic; therefore, more advanced clustering algorithms are needed for its classification.

Figure 12 shows the precision (8) of classification obtained by the AutoClass algorithm for every play-out algorithm separately. The classification of the traffic generated by *MMS* achieves the highest precision for all the examined clustering algorithms. Depending on the network conditions, between 66% and 72% of the play-out strategies classified as *MMS* are actually *MMS*. The second score achieves the *Festive* strategy while *PANDA* and *LOLY* are little harder to classify precisely Similarly to the accuracy, the best score is achieved for the scenario, where the majority of the players remain in a mobile environment. With the stabilisation of network conditions, the precision of classification decreases.

7 Conclusion

In this work, using the cluster analysis and information about traffic flows gathered at the application level, we were able to identify groups of traffic flows that were generated by four exemplary play-out strategies. As the research shows, the performance of the clustering algorithms differs depending on the applied parameters and network environment of the video players. Generally, unstable network conditions like variable throughput and packets delays encountered, e.g. in mobile networks, exposes a weakness of the play-out algorithms

which tend to make similar decisions about adaptations to the current network conditions. The algorithms make more quality switches which quite often are synchronised. Although such behaviour has a negative impact on users' QoE, it generates more correlated data what increases the probability of the correct identification of video flows.

Depending on the applied parameters, the performance of the examined algorithms, namely MinMax K-Means, OPTICS and AutoClass, is roughly similar. However, AutoClass have a slight advantage in case of mobile networks characterised by variable conditions, while MinMax K-Means achieves a better score for stable network environments. Probably, this may be explained by how the algorithms build their clusters. As it was mentioned in Section 4, the clusters defined by MinMax K-Means have fixed size and are symmetrical, while OPTICS and AutoClass build their clusters with more flexibility. In the case of stable network conditions, the shape of the clusters probably matches better these defined by MinMax K-Means. However, this assumption requires further research. In general, the results are in agreement with the other works in which AutoClass obtains usually the best score [19, 20].

The easiest to classify is the traffic which is produced by relatively simple play-out strategy, e.g. *MMS* or *Festive*. The adaptive play-out strategies of more advanced algorithms try to avoid mutual synchronisation of streams quality, therefore, their traffic is less correlated and harder to classify.

It is worth mentioning that the results of obtained by our approach may be boosted by gathering and taking into account additional information about the network traffic, e.g. video segment sizes or patterns of communications between a video player and a server. This information could be used to create hybrid classification models supported by supervised and unsupervised learning processes. Thus, in our future works, we plan to apply a combination of several specialized classifiers which are to be trained with the same data. An incoming input is to be evaluated by all the classifiers and the obtained results should be merged by means of a combination method, such as e.g. maximum likelihood. Thus, each of the applied classifiers should vote on one class, and the class with the most votes should be the output.

Acknowledgements The work was carried out within the statutory research project of the Institute of Informatics, BK-213/RAU2/2018, Gliwice, Poland.

References

1. Alcock S, Nelson R (2011) Application flow control in YouTube video streams. ACM SIGCOMM Comput Commun Rev 41(2):24–30
2. Alshammari R, Zincir-Heywood AN (2011) Can encrypted traffic be identified without port numbers IP addresses and payload inspection? Comput Netw 55(6):1326–1350
3. Amancio DR, Comin CH, Casanova D, Travieso G, Bruno OM, Rodrigues FA, da Fontoura Costa L (2014) A systematic comparison of supervised classifiers. PloS one 9(4):e94137

4. Ankerst M, Breunig MM, Kriegel H-P, Sander J (1999) OPTICS: ordering points to identify the clustering structure. In: ACM sigmod record, vol 28. ACM, pp 49–60
5. Apache Apache Web Server, www.apache.org
6. Arndt DJ, Zincir-Heywood AN (2011) A comparison of three machine learning techniques for encrypted network traffic analysis. In: 2011 IEEE symposium on computational intelligence for security and defense applications (CISDA). IEEE, pp 107–114
7. Bakhshi T, Ghita B (2015) User traffic profiling. In: Internet technologies and applications (ITA), 2015. IEEE, pp 91–97
8. Biernacki A (2016) Analysis of aggregated HTTP-based video traffic. J Commun Netw 18(5):826–836
9. Biernacki A (2017) Improving Video Quality by Diversification of Adaptive Streaming Strategies. KSII Trans Internet Inf Syst 11(1):374–395
10. Botta A, Dainotti A, Pescapè A (2012) A tool for the generation of realistic network workload for emerging networking scenarios. Comput Netw 56(15):3531–3547
11. Bujlow T, Carela-Español V, Barlet-Ros P (2015) Independent comparison of popular DPI tools for traffic classification. Comput Netw 76:75–89
12. Callado A, Kamienski C, Szabó G, Gero BP, Kelner J, Fernandes S, Sadok D (2009) A survey on internet traffic identification. IEEE Commun Survey Tutorial 11(3):56–76
13. Cao Z, Xiong G, Zhao Y, Li Z, Guo L (2014) A survey on encrypted traffic classification. In: International conference on applications and techniques in information security. Springer, pp 73–81
14. Cisco (2017) Cisco visual networking index: forecast and methodology, 2016-2021. Technical report
15. Datta J, Kataria N, Hubballi N (2015) Network traffic classification in encrypted environment: a case study of google hangout. In: 2015 21st national conference on communications (NCC). IEEE, pp 1–6
16. Du Y, Ruhui Z (2013) Design of a method for encrypted P2p traffic identification using K-means algorithm. Telecommun Syst 53(1):163–168
17. Dubin R, Dvir A, Pele O, Hadar O, Richman I, Trabelsi O (2016) Real Time Video Quality Representation Classification of Encrypted HTTP Adaptive Video Streaming-the Case of Safari. arXiv: 1602.00489
18. Dubin R, Hadar O, Dvir A, Pele O (2018) Video Quality Representation Classification of Encrypted HTTP Adaptive Video Streaming. KSII Trans Internet Inf Syst 12(8):3804–3819
19. Erman J, Arlitt M, Mahanti A (2006) Traffic classification using clustering algorithms. In: Proceedings of the 2006 SIGCOMM workshop on mining network data. ACM, pp 281–286
20. Erman J, Mahanti A, Arlitt M (2006) Internet traffic identification using machine learning. In: 2006 IEEE global telecommunications conference, GLOBECOM'06. IEEE, pp 1–6
21. Ester M, Kriegel H-P, Sander J, Xu X et al (1996) A Density-based algorithm for discovering clusters in large spatial databases with noise. In: Kdd, vol 96, pp 226–231
22. Famaey J, Latré S, Bouten N, Van de Meerssche W, De Vleeschauwer B, Van Leekwijck W, De Turck F (2013) On the merits of SVC-based HTTP adaptive streaming. In: 2013 IFIP/IEEE international symposium on integrated network management (IM 2013). IEEE, pp 419–426
23. Finsterbusch M, Richter C, Rocha E, Muller J-A, Hanssgen K (2014) A survey of payload-based traffic classification approaches. IEEE Commun Survey Tutorial 16(2):1135–1156
24. Gámez RCL, Martí P, Velasco M, Fuertes JM (2006) Wireless network delay estimation for time-sensitive applications. Automatic Control Department, Technical University of Catalonia Research report ESAII RR-06-12
25. Hemminger S (2005) Network emulation with NetEm. In: Linux conference au, pp 18–23
26. Hooghiemstra G, Van Mieghem P (2001) Delay distributions on fixed internet paths. Technical report, Delft University of Technology
27. Jacobson V, Leres C, McCanne S (2015) TCPDUMP public repository. Web page at http://www.tcpdump.org
28. Jiang J, Sekar V, Zhang H (2012) Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. In: Proceedings of the 8th international conference on emerging networking experiments and technologies. ACM, pp 97–108
29. Keogh E, Kasetty S (2003) On the need for time series data mining benchmarks: a survey and empirical demonstration. Data Min Knowl Disc 7(4):349–371
30. Kim H, Claffy KC, Fomenkov M, Barman D, Faloutsos M, Lee K (2008) Internet traffic classification demystified: myths, caveats, and the best practices. In: Proceedings of the 2008 ACM CoNEXT conference. ACM, pp 11
31. Kwapień J, Drożdż S (2012) Physical approach to complex systems. Phys Rep 515:116–225
32. Lederer S, Müller C, Timmerer C (2012) Dynamic adaptive streaming over HTTP dataset. In: Proceedings of the 3rd multimedia systems conference, pp 89–94
33. Li Z, Zhu X, Gahm J, Pan R, Hu H, Begen AC, Oran D (2014) Probe and adapt: rate adaptation for http video streaming at scale. IEEE J Sel Areas Commun 32(4):719–733

34. Mantegna RN (1999) Hierarchical structure in financial markets. European Phys J B 11(1):193–197
35. Miller K, Quacchio E, Gennari G, Wolisz A (2012) Adaptation algorithm for adaptive streaming over HTTP. In: 2012 19th international packet video workshop (PV), pp 173–178
36. Müller C, Timmerer C (2011) A VLC media player plugin enabling dynamic adaptive streaming over HTTP. In: Proceedings of the 19th ACM international conference on multimedia, pp 723–726
37. Nguyen TTT, Armitage G (2008) A survey of techniques for internet traffic classification using machine learning. IEEE Commun Survey Tutorial 10(4):56–76
38. Pacheco F, Exposito E, Gineste M, Baudoin C, Aguilar J (2018) Towards the deployment of machine learning solutions in network traffic classification: a systematic survey. IEEE Commun Survey Tutorial 1–1. https://ieeexplore.ieee.org/document/8543584
39. Sandoval Jr. L (2013) Cluster formation and evolution in networks of financial market indices. Algorithmic Finance 2(1):3–43
40. Singh H (2015) Performance analysis of unsupervised machine learning techniques for network traffic classification. In: 2015 5th international conference on advanced computing & communication technologies (ACCT). IEEE, pp 401–404
41. Sodagar I (2011) The mpeg-dash standard for multimedia streaming over the internet. Multimed IEEE 18(4):62–67
42. Stockhammer T (2011) Dynamic adaptive streaming over HTTP: standards and design principles. In: Proceedings of the 2nd annual ACM conference on multimedia systems, pp 133–144
43. Taylor VF, Spolaor R, Conti M, Martinovic I (2016) Appscanner: automatic fingerprinting of smartphone apps from encrypted network traffic. In: 2016 IEEE European symposium on security and privacy (EuroS&P). IEEE, pp 439–454
44. Tzortzis G, Likas A (2014) The MinMax k-Means clustering algorithm. Pattern Recogn 47(7):2505–2516
45. Velan P, Čermák M, Čeleda P, Drašar M (2015) A survey of methods for encrypted traffic classification and analysis. Int J Netw Manag 25(5):355–374
46. Yao J, Kanhere SS, Hossain I, Hassan M (2011) Empirical evaluation of HTTP adaptive streaming under vehicular mobility. In: International conference on research in networking. Springer, pp 92–105
47. Zhang J, Chen X, Xiang Y, Zhou W, Wu J (2015) Robust network traffic classification. IEEE/ACM Trans Netw 23(4):1257–1270
48. Zhang J, Xiang Y, Wang Y, Zhou W, Xiang Y, Guan Y (2013) Network traffic classification using correlation information. IEEE Trans Parallel Distrib Syst 24(1):104–117

Arkadiusz Biernacki received the M.Sc. and Ph.D degree in Computer Science from the Silesian University of Technology, Poland, in 2002 and Ph.D. 2007 respectively. From 2007 he is an Assistant Professor at the Silesian University of Technology. His research interests focus on network traffic modelling and computer system simulations.