Improving Video Quality by Diversification of Adaptive Streaming Strategies

A.Biernacki
Institute of Computer Science, Silesian University of Technology,
Akademicka 16, 44-100 Gliwice, Poland
E-mail: arkadiusz.biernacki@polsl.pl

Received May 4, 2016; revised September 12, 2016; accepted October 12, 2016; published January 31, 2017

Abstract

Users quite often experience volatile channel conditions which negatively influence multimedia transmission. HTTP adaptive streaming has emerged as a new promising technology where the video quality can be adjusted to variable network conditions. Nevertheless, the new technology does not remain without drawbacks. As it has been observed, multiple video players sharing the same network link have often problems with achieving good efficiency and stability of play-out due to a mutual interference and competition among video players.

Our investigation indicates that there may be another cause for under-performance of the streamed video. In an emulated environment, we implemented three algorithms of adaptive video play-out based on bandwidth or buffer assessment. As we show, traffic generated by players employing the same or similar play-out strategies is positively correlated and synchronised (clustered), whereas traffic originated from different play-out strategies shows negative or no correlations. However, when some of the parameters of the play-out strategies are randomised, the correlation and synchronisation diminish what has a positive impact on the smoothness of the traffic and on the video quality perceived by end users. Our research shows that non-correlated traffic flows generated by play-out strategies improve efficiency and stability of streamed adaptive video.

***Keywords*:** Network traffic; Adaptive video; Multimedia communication; Video streaming; Data mining

https://doi.org/10.3837/tiis.2017.01.020 **ISSN : 1976-7277**

1. Introduction

During the past years, web-based video sharing services like YouTube, Hulu or Dailymotion have become very popular. The users of YouTube, which allows for the distribution of user-produced multimedia content, alone request 6 billion hours of video every month [1]. Consequently, such popularity results in a drastic shift in Internet traffic statistic and leads to an increase of traffic generated by web-based video sharing services. It is estimated that this type of traffic will account for about 80% to 90% of the global Internet traffic in 2019 according to the report published by Cisco [2].

Video streaming in the above-mentioned services is HTTP-based; therefore, being transported using TCP. HTTP and TCP are general purpose protocols and were not primarily designed for streaming of multimedia. Thus, attempts are being made to adapt the delivery of multimedia content to the Internet environment. One of such attempts tries to introduce an additional layer of application control to transmitted video traffic [3]. Since TCP is designed to deliver data at the highest available transmission rate, it may sometimes be reasonable for a sender to provide additional flow control if it is not necessary for application data to reach a receiver as fast as TCP would otherwise allow. Therefore, the application may limit the rate at which data is passed to a network stack for transmission producing ON-OFF cycles, where during the ON time a block of data is transferred at the end-to-end available bandwidth that can be used by TCP, and during the OFF time, the TCP connection remains idle.

Furthermore, the 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, based on a state of a video player, for example on the length of the player buffer, or on the state of a network environment, for example on an amount of available bandwidth [3]. In the current approaches, usually the video player alone chooses suitable video quality and is responsible for the adaptation to a 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. The players keep information about their buffer state only to themselves, therefore, there is no coordination among clients. 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 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**. Some research also shows that when more video players share the same bottleneck link, due to the temporal overlap of their ON-OFF periods, the players may overestimate or underestimate the available bandwidth. This inaccurate assessment negatively impacts on streamed video as the players repeatedly switch between different quality levels and inefficiently use available network bandwidth [4].

In our work, we show that there is another possible cause of inefficiency and instability. 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. However, in the case when the play-out strategies are different, or at least, have different parameters, the positive correlation diminishes. Furthermore, with the decreasing correlation, the aggregated traffic is

less variable and oscillates in a tighter range what has a direct impact on perceived video quality. As our examination shows, less variable traffic improves efficiency and stability of video streaming.

Our approach requires only very limited modification of play-out algorithms and basically no server-side support. We manipulate with values of some basic parameters of adaptive algorithms and we try to estimate how they influence the characteristic of the network traffic. In the contrast to the popular techniques proposed, for example in [5], we do not operate on the level of aggregate traffic, but on the level of individual flows. Thus, to a certain extent, instead of dealing with the effects of variability, we try to prevent its cause. For this purpose, we investigate inter-dependencies among traffic flows, i.e. we search for correlation in the traffic generated by particular video players. Moreover, we inspect visually traffic traces and compute popular traffic characteristics like variability or the long range dependence (LRD) parameter. Finally, we analyse how the randomisation of adaptive algorithms parameters impact the quality of video play-out. The above analysis and their results are the main contributions of our work.

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 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 three different play-out algorithms. Two of them mimic the behaviour of real commercial video players and the other mimics a popular play-out strategy described in the literature.

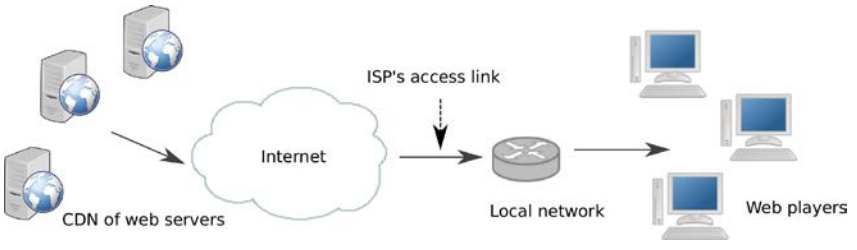


Fig. 1. HTTP streaming scenario. Video traffic is multiplexed on the ISP access link.

The rest of the paper is organised as follows: in the next section, we describe algorithms engaged in a traffic flow control at the application. Then, we review other related works to our proposition. After that, we present the methodology used in our experiments. In the further two chapters, we compare an interdependence among traffic flows and statistics of the whole aggregated traffic for two scenarios where the adaptive algorithms have fixed and randomised parameters values. Finally, we discuss how the diversification of parameters values influences the quality of streamed video.

2. Application level flow control

In most modern video systems based on HTTP, the video file is divided into chunks of fixed time and the server pushes them sequentially to the client at a rate little higher than the video-bit rate of the transmitted content. As a result, the transmitted traffic creates an ON-OFF pattern, where ON and OFF periods have a constant length [6]. 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. This technique was adopted by Dynamic Adaptive Streaming over HTTP (DASH), which is a MPEG standard pursuing the interoperability between devices and servers of various vendors [3][7]. 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 transmitted traffic also creates the ON-OFF pattern, but this time, the duration of the ON and OFF periods is not constant anymore but is a discrete random variable, whose values are dependent on the logic of the play-out algorithm.

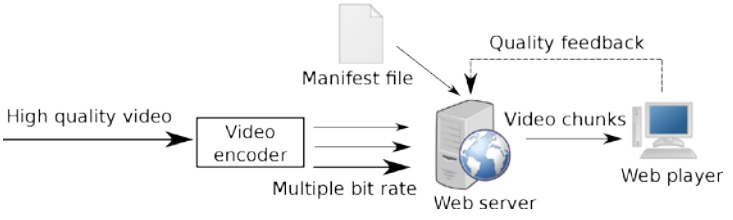


Fig. 2. Architecture of a video adaptive system based on HTTP.

The stream-switching technique is employed today less or more in some proprietary video players, among others in Apple HTTP-based streaming [8], Microsoft IIS Smooth Streaming [9] or Adobe Dynamic Streaming [10].

As the above adaptive video systems do not share many details about the employed algorithms, in our experiments we use the template of an adaptive streaming algorithm based on a bandwidth estimation which is implemented in the open-source software described in [11]. Moreover, we implemented two further adaptive algorithms which extend the idea proposed in the template, and apart from the network throughput, they take into account also the level of a player buffer or add some more elaborate features like throughput prediction. The implemented hybrid strategies are employed in popular software [12] or, at least, gained a fair amount of publicity [13].

2.1 Play-out algorithms

The bandwidth estimation algorithm tries to adjust a bit-rate of video to the measured network throughput. It is employed by DASH plug-in which is a part of VLC media player [11]. The approach is relatively simple and its main points were summarised in **Algorithm 1**. The algorithm calls a function which measures average network throughput n_x in a certain time window ΔT. This window can be considered as a parameter of the algorithm and it may be stretched or shortened in order to optimise streamed video quality. When the video bit-rate v, which is needed for a smooth video play-out, is lower than the computed average network

throughput n_x reduced by ΔL, **line 1**, the algorithm reports that the video quality level q should be increased, i.e. the chunk download module asks the server for bigger chunks, encoded in higher quality. When the throughput is not sufficient for the given level of video quality, **line 4**, the opposite situation takes place: the quality level q is decreased and the download module is instructed to obtain chunks of poorer quality what simultaneously demands less network throughput. The parameter ΔL marks a region of network throughput for which there is no need to switch the quality to a higher level. As a result, the parameter plays a stabilising role and prevents switching the quality levels too frequently, what could have a negative impact on the overall video quality perceived by users. The constants $Q_{\min}$ and $Q_{\max}$ define a range of available levels of the quality.

Input: q – current video quality level
v – video bit rate
ΔT – time window for measurement of network throughput
n_x – measured network bandwidth in the period ΔT
1: **if** $v < n_x - \Delta L$ **then**
2: $q \leftarrow q + 1$
3: **end if**
4: **if** $v > n_x$ **then**
5: $q \leftarrow q - 1$
6: **end if**
Ensure: $q \in \{Q_{\min}, \ldots, Q_{\max}\}$
7: **return** q

Algorithm 1. Adaptation based on a bandwidth estimation

Algorithm 1 has some problems related to accurate bandwidth estimation and stability. Therefore, in order to overcome the drawbacks, the algorithms proposed in the literature base their estimation additionally on the level of a player buffer and on more elaborate assessment of network throughput using for this purpose, for example, prediction techniques. We implemented two such strategies presented in literature in the recent years.

The first implementation, and our second test algorithm, uses the Microsoft Smooth Streaming (MSS) algorithm, which is based on the open source version of the algorithm of the MSS video player and is extensively described in [12].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 heuristic, and our third test algorithm, relies on the *PANDA* (Probe AND Adapt) algorithm proposed in [13]. 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.

We examined the multiplexed traffic generated by the above-mentioned algorithms taking into account two scenarios. In the first scenario, the algorithms have the same, fixed parameters, i.e. video players employ exact copies of the three respective algorithms. In the second scenario, we assign to some of the algorithms parameters uniformly distributed random values chosen from a constrained set. Thus, in this case, the players employ varied strategies. The list of the parameters with their fixed and randomised values is presented in **Table 1**. The

intervals are set as the minimum and maximum parameters values, for which we did not observe a significant degradation in performance of the examined algorithms during the laboratory experiments.

Table 1. Default values and ranges of randomisation for the selected parameters of the analysed algorithms.

Sym.	Algorithm	Parameters		
		Description	**Default**	**Randomised**
A	**Algorithm 1**	ΔL – region of network throughput for which there is no need to switch the quality to a higher level;	$25\%n_x$	$15\%\text{-}75\%n_x$
		ΔT – time window for measurement of network throughput	45 s	15-90s
B	MSS [12], Algorithm 1	P – buffer state;	12 s	8-24s
		time window for measurement of network throughput	20 s	14-60 s
C	PANDA [13], Algorithm 2	Δ – quantization margin;	$25\%n_x$	$15\%\text{-}75\%n_x$
		moving average of past *n* measurements for the estimation of network throughput	10	4-16

3. Related works

Some researchers observed that multiple video players compete among themselves for available throughput needed to transfer video chunks. The authors of [14] show that inaccurate measurement of received data results in instabilities of a video play-out and a degradation of video quality. They attribute the responsibility for this issue to the ON-OFF pattern of traffic produced by the application layer: even if one video player obtains its fair share sufficiently during the ON period, it can fail to correctly estimate available bandwidth due to the overlapped ON periods with the other player. As a consequence, the former player will switch to a lower rate than the network conditions allow, and network throughput remains underutilised. The authors of [15] examined a video player competing against another video player, and against a long-lived TCP flow. Interestingly, they demonstrated that inaccurate throughput estimation occurs even when the competing flow does not exhibit the ON-OFF behaviour. Therefore, some research works, e.g. [16], try to improve the algorithm by predicting the future bandwidth, while the others, e.g. [17][18][19], propose an algorithm based on measurement of buffer occupancy. Nevertheless, the buffer reactive algorithms perform fine in many cases, but sometimes they have a tendency to too frequent oscillation between video bit rates [17]. Thus, there are concepts of more sophisticated buffer management, e.g. in [20] the authors present an algorithm which eliminates ON-OFF periods from the traffic through two buffer management states. The authors claim that the proposed scheme improves fairness in the system by 45% compared to the conventional adaptive player.

Some of the authors propose more elaborate strategies, e.g. FESTIVE [21] or PANDA [13] play-out strategies, which take not only network bandwidth or a player buffer into account but also consider stability, efficiency and fairness. Additionally, FESTIVE includes an element of randomness – its scheduler ensures that the request time of a player is independent of its start time. In [22] the so-called network control plane, designed to take into account

scalability and adaptiveness issues, is placed on top of the controlled network. The plane co-operates with distributed buffer-based adaptation techniques implemented in clients but does not interact with a video server.

Some researchers see a chance of overcoming the problems with the bit-rate instabilities and an effective usage of throughput by an engagement of server-side mechanisms. For example, in [23] the authors suggested a traffic shaping method, implemented at home gateways, to reduce an extent of observed instability and unfairness among competing video players. Another proposition is a server-based method of traffic shaping that can reduce oscillations of a video bit-rate received by a player [24]. Liu et al. in [25] follow a similar approach where the rate is shaped according to the maximisation of a QoE metrics. The authors of [26] presented a method of video pacing that reduces unnecessary traffic and simultaneously conserves earlier video quality. In [27], the authors suggested to dynamically adjusts a segment size of TCP and a number of video streams in order to optimize throughput of a connection. Another proposal is to locate a traffic shaper between a video server and users' players in order to adaptively transform a video bit-rate to current network conditions; however, this proposition requires a feedback from the players [28]. Finally, in [29] the authors employ a complex strategy in which the use in-network quality optimization agents monitoring the available throughput using sampling-based measurement techniques and optimize QoE of each client. This in-network optimization is achieved by applying centralized as well as distributed algorithms what requires coordination between a server and clients.

Taking into account the above issues with fairness, efficiency and stability, one of the possible approach to a video quality improvement is to focus on the traffic engineering methods which may to some extent alleviate the mentioned problems with competition among video players and correct bandwidth estimation. Though the transfer rate control implemented at the application level in video adaptive systems may appear analogous to the TCP congestion control, there are some key differences between these two logics, which may have an impact on a traffic characteristic: the two control algorithms operate at different levels in the protocol stack; TCP is a connection-oriented protocol while video adaptation is a connectionless protocol and TCP operates at the packet level whose size is about 1 KB, while video system operates at the level of segments, whose size is usually hundreds of kilobytes. Nevertheless, the amount of work focusing on traffic characteristics generated by adaptive video systems and its impact on video quality is quite limited. Some of the works tried to investigate details of traffic characteristics, for example in [30] the authors the examine the periodic behaviour of traffic generated by Microsoft Smooth Streaming and Netflix. In [31] the authors provide formulas for average intensity of network traffic and its variability. However, the proposed analytical model takes into account only non-adaptive streaming. Variability of non-adaptive video traffic based on HTTP in small timescales (below 1 ms) and methods to eliminate this undesirable phenomenon is studied also in [32]. To our best knowledge, none of the work has been focused on an investigation of how video play-out algorithms impact the traffic characteristics and how this characteristics influences on the quality of video.

4. Laboratory set-up

In order to capture traffic traces, which were generated by an adaptive video system, we prepared a test environment emulating content distribution network (CDN), which in a 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. 3**.

As the NEE, we used a network emulation node based on the built-in Linux Kernel module *netem* [33]. The module is capable of altering the network QoS parameters such as network bandwidth or packet delays; thus, it allows for testing data transmission in different network environments.

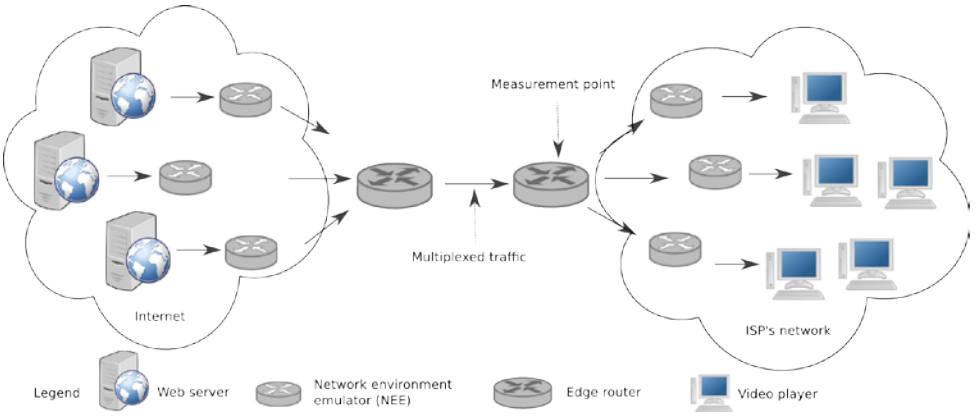


Fig. 3. An emulated CDN network which consist of network environment emulators, web servers, video players and a measurement point located in an edge router.

The role of the web server plays Apache [34], 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 [35][36] 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 [2] 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 [37].

As a video player, we chose the VLC media player with the DASH plug-in [11]. 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 **Algorithm 1** and two other algorithms acquired from literature. 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 the account the findings presented in [38], the content was divided into 2 s length segments and the size of the player buffer was set to 6 s. Similarly to the server-side, on the client-side packet delays have the Weibull distribution while the bandwidth is distributed uniformly. **Table 2** summarises the basic parameters of the experiment.

From several video files, acquired from [39] and presented in **Table 3**, every player repeatedly chose a random file to download. The duration of each of the experiment was set to twenty minutes. Using the edge router with installed capturing software, based on Tcpdump

and Libcap [40], we obtained aggregated traffic traces. These traces were converted into time series represented by a point process. As we were interested in the properties of relatively short time traces of about several minutes, we did not introduce users' churn or take into account switching of the players between different CDN servers. For the same reason, we also did not take into account video length distribution, its popularity or the patterns of users' access, which could possibly lead to a more complex model of aggregated video traffic in a macro-scale. We repeated the experiment five times and averaged the results of the analysis.

Table 2. Simulation parameters

Parameter	Value(s)
Number of webservers	3
Number of NEE	6
Number of video players	120
Bandwidth at the edge routers	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	2 s
Buffer size of players	6 s

Table 3. Video clips used during the experiments.

Name	Genre	Bit-rate 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	

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, and manageability of a simulation while preserving the support for real video adaptive applications.

5 Analysis of traffic correlations

In the last decade, much of research focused on the investigation of statistical properties of natural or social systems. Some of the works showed that the correlation between different entities belonging to a system and topological characteristics of links connecting these entities cannot be described in terms of random graphs. For example, the distribution of a degree

of nodes in graphs describing entities in the World Wide Web, the Internet, or social networks is power-law rather than Poisson-type, and cannot be constructed by randomly connecting previously unconnected pairs of nodes until a predefined total number of edges in the network is reached [41][42]. Taking into account heterogeneity of players in video systems and their adaptiveness, the analysis of the interaction between these entities may allow achieving a broader view of the nature of the system and contributing to its better understanding. Furthermore, the analysis may be a foundation for developing physical models which will be able to capture more details of network traffic compared to purely mathematical models describing aggregated traffic traces.

In the first step of our analysis, we are interested in a correlation of traffic generated by particular video players. While adjusting video bit-rate to a current network environment, each of the players generates time series representing respective quality levels q_i (or corresponding video bit-rates), see also **Algoritm 1**, which presents examples of negative and positive correlation among particular players. From the above set of time series, we computed the correlation coefficient between any pair of players as

$$\rho_{ij} = \frac{\langle Q_i Q_j \rangle - \langle Q_i \rangle \langle Q_j \rangle}{(\langle Q_i^2 \rangle - \langle Q_i \rangle^2)(\langle Q_j^2 \rangle - \langle Q_j \rangle^2)}, \tag{1}$$

where $Q_i, Q_j \in \{Q_{\min}, \ldots, Q_{\max}\}$ represent a stochastic process whose elements are quality levels of video 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 positive correlation between singular flows will increase the variability of aggregated traffic. Single traffic flows can be considered synchronised and will simultaneously increase and decrease their intensity. However, the increase will be confined to the network bandwidth – when the aggregated traffic reaches the network capacity, the players will simultaneously decrease the quality. The decrease will be restrained by the play-out algorithms, which will report that higher network throughput is available than currently being utilised or that their buffer level allows increasing the quality. Thus, the flows will simultaneously start increasing their intensity, and the above pattern will repeat. Such synchronous flows lead to oscillations of the aggregated traffic which are responsible for packet losses and a decrease of network throughput [43].

In the case of the negative correlations, an increase of one flow intensity is compensated by a decrease of the other flow intensity. Thus, the aggregated network traffic should remain more or less at the similar level preserving its smoothness, what is desirable from a traffic engineering point of view.

5.1 Analysis of the cross-correlation matrix

The correlation coefficient ρ_{ij} (1) is computed between all the possible pairs of players downloading video from the emulated CDN. The goal of the present study is to obtain the taxonomy of the play-out algorithms using only the time series representing the quality of streamed video. 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. In order to present such a large correlation matrix, we averaged the correlation coefficient between the analysed groups.

The graphical representation of the cross-correlation matrix for groups of players engaged in the experiment is presented in **Figs. 5** and **6**, where the darker the specified region, the higher the absolute value of the averaged correlation coefficients ρ_{ij}; and, respectively, the lighter the specified region, the absolute value of ρ_{ij} is lower. Taking into account that we examine correlations between the quality levels of video players employing three different play-out algorithms, to facilitate the analysis, we divided the matrix into nine regions. Every region represents the dependency between groups of players using a particular play-out algorithm. As the matrix is symmetrical, it is sufficient to analyse only half from nine regions.

In order to enhance visibility, we analyse negative, see **Fig. 5**, and positive correlations, see **Fig. 6**, separately. The regions laid on the diagonal of the matrix show correlation between groups of players employing the same algorithm. Furthermore, the analysis takes into account two scenarios denoted as *Fixed* and *Random*. The set-up of both scenarios is similar and consists of three groups of players, each counting 40 players, employing its play-out strategy and downloading simultaneously video content. However, in the former scenario, the respective play-out algorithms are exact copies of each other, i.e. their parameter values are the same. In the latter scenario, the parameter values of the respective algorithms are randomly selected as presented in **Table 1**. Thus, with great probability, each player has its own unique play-out strategy.

We may notice, that there are some negative dependencies introduced by *PANDA* strategy in the relation to the bandwidth-based and *MSS* algorithms, as presented in **Fig. 5A**. When the parameters of the play-out strategies have random values, the negative correlation between *PANDA* and *MSS* strategies disappears. The negative dependencies may be probably explained by the internal characteristics of the examined strategies. The strategies presented in **Algorithm 1** and *MSS* include common elements of estimation of network throughput based on direct measurement. However, the *PANDA* strategy is based not on direct measurement of the network throughput, but instead, it tries to estimate more accurately network conditions by eliminating the influence of throughput fluctuations, see **Algorithm 2** in [13].

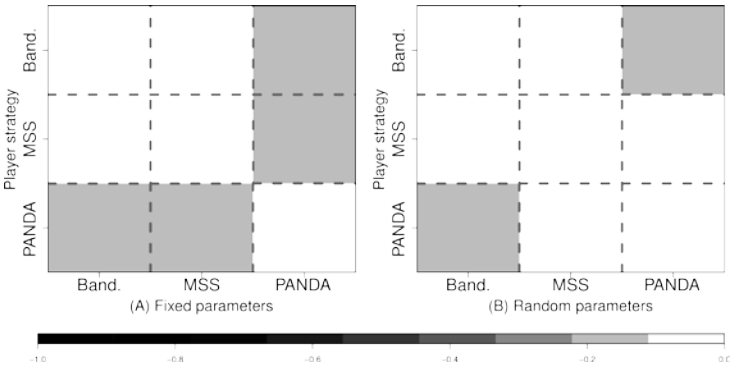


Fig. 5. Averaged negative correlations for 120 video players.

Table 1 presents a legend for the algorithm symbols.

The above interpretation becomes even more evident when we start to analyse positive de-

pendencies, which are even more substantial. Pure bandwidth based and *MSS* strategies are mutually positively correlated. Thus, we may confirm the observation made during the analysis of the negative correlations – the strategies sharing common ideas tend to be positively correlated. Therefore, the pure bandwidth-based estimation strategy, presented in **Algorithm 1**, is correlated with the *MSS* strategy and, rather weakly, with *PANDA* strategy, what was denoted in **Fig. 6A**. After randomisation of the algorithms parameters, the positive correlation is significantly weaker for all pairs of the play-out algorithms, as presented in **Fig. 6B**. In the case of *PANDA* strategy, the positive correlation ceases to exist at all.

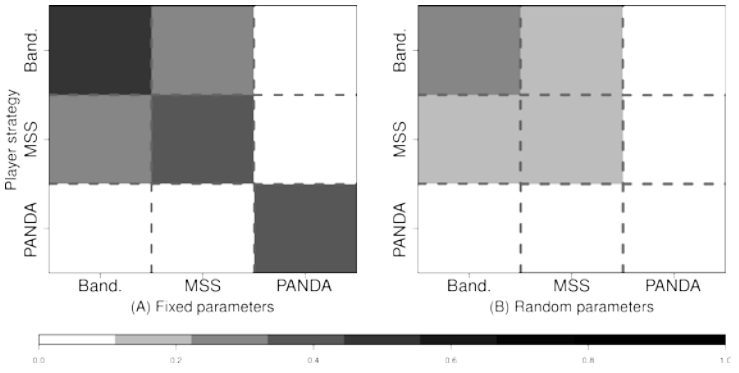


Fig. 6. Averaged positive correlations for 120 video players.

5.2 Analysis of clusters

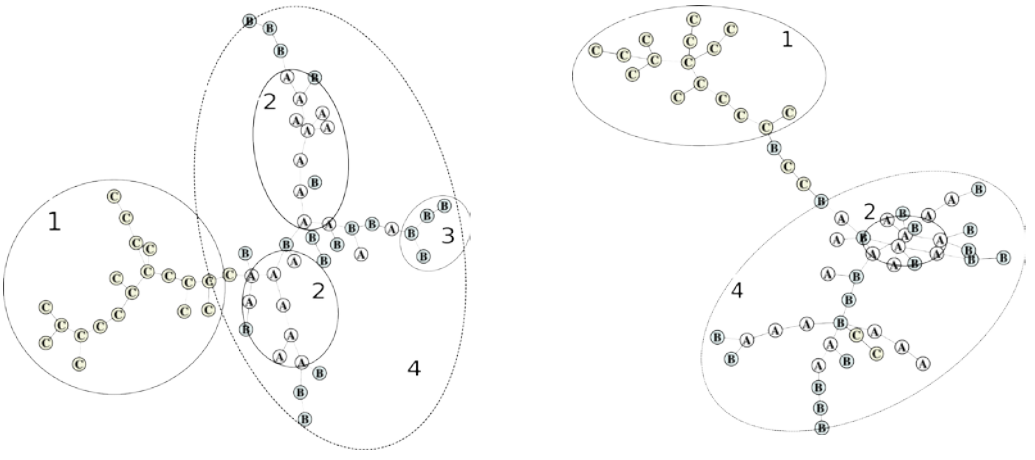
The cross-correlation matrix is a common way to investigate the interaction between entities of the system. However, as any statistical estimator, the sample correlation matrix is unavoidably affected by statistical uncertainty due to the finite size of the sample. In order to mitigate this nuisance, one needs to apply filtering methods which are able to remove from the correlation matrix, at least, part of the noise. From the abundance of different methods dedicated for this purpose, we chose the minimum spanning tree (MST) algorithm, which is relatively simple to describe and allows to get an insight into the most important relations in the examined system. Using the correlation matrix defined from the coefficients in (1), we can obtain a fully connected graph with vertices corresponding to the video players and edges weight corresponding to the correlation coefficients. Then applying the MST algorithm, we can eliminate the less relevant information by removing the weakest edges.

The MST algorithm operates on a concept of a distance identified as edges weight. Given a connected and undirected graph, a spanning tree of that graph is a subgraph that is a tree and connects all the vertices together. After assigning a weight to each edge, the MST algorithm computes the sum of the weights of the edges and chooses the tree with a sum of the weights less than or equal to the weight of every other spanning tree. However, we cannot build the graph directly from the correlation matrix because its elements in their original form are not suitable for a weight of the graph edges, for example a distance between the same two elements would have been one ($\rho_{ij} = 1$) 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 [44] we use the transformation

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

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 distance matrix **D**, composed from elements defined in (2), is then used to determine the MST connecting the bit-rate requested by n players. When we assign a weight to each edge which represents the distance d_{ij}, we can compute a weight of the spanning tree by computing the sum of the weights of the edges in that spanning tree. A spanning tree is then a graph without loops connecting all the n vertices with $n-1$ edges. As the original fully connected graph is metric with distance d_{ij} which is decreasing with ρ_{ij}, therefore the MST algorithm selects the $n-1$ stronger (i.e. shorter) links which span all the nodes. For the generation of the MST we applied Prim's algorithm [45] implemented as *minimum.spanning.tree* function in the R statistical environment [46].



(a) (b)

Fig. 7. Fragment of a MST connecting 120 video players for adaptive algorithm with parameters set to a) fixed, b) random values. Regions 1, 2 and 3 present clusters consisting of more than four players employing the same play-out strategies. Region 4 presents a cluster formed by players employing different strategies. Table 1 presents a legend for the algorithm symbols.

The MST for our analysis is presented in **Fig. 7**. Players employing the same play-out algorithm are represented as nodes of the same colour and symbol, see also **Table 1**. In the case, when all values of the play-out algorithms parameters are set to fixed values, see **Fig. 7a**, we can infer that most of its parts are composed of vertices representing the same play-out strategies. Other words, players employing the same strategy tend to form clusters. Larger clusters are formed by the players using *PANDA* strategy what was denoted as Region 1. in The strategies described in **Algorithm 1** and MSS form smaller clusters denoted as Region 2 and 3 respectively. However, as the two above strategies share a common idea (non-filtered bandwidth estimation), therefore players employing these strategies form also one bigger cluster denoted as Region 4. Such a connection indicates that sometimes there can be a strong correlation even between players implementing different strategies. These dependencies are not directly captured by the correlation analysis, presented in **Figs. 5** and **6**, as it shows only averaged results.

After assigning random values to the parameters of the adaptive strategies, the monolithic

group consisting of players implementing *PANDA* strategy is split into three smaller clusters, see **Fig.7b**. Also, the randomisation reduces larger groups of players implementing bandwidth-based and *MSS* strategies. The decrease of a cluster size results in a lower correlation and synchronisation among traffic flows.

Summarising the above dependency analysis, there is a clear correlation between the quality, and thereby also the amount of traffic received by video players. Furthermore, the positive correlation can be also observed, when the play-out algorithms even partially base their decisions on the same ideas, for example on the network throughput estimation. The negative correlation is visible when there are clear differences in methodology of throughput estimation, for example direct measurement versus prediction. One of the possible explanations for this dependence is that when the predictor correctly estimates future network bandwidth, transmission generated by *PANDA* algorithm fills the recess left by the two other algorithms. Thus, not a simple direct measurement, but a more advanced estimation of network bandwidth allows *PANDA* to escape from synchronisation bounds which are shared by the other algorithms. However, as the multiple players share common network path and employ *PANDA* approach, consequently their algorithms will provide quite a similar prediction. Therefore, a certain level of positive synchronisation among *PANDA* players in inevitable.

6 Analysis of aggregated traffic

The packets of the different video flows are usually placed on a single physical link – for example, a packet is transmitted for one connection, then a packet for a second, another for the first, then two packets in a row for a third, and so forth. This intermingling is referred to as "statistical multiplexing" in the packet network literature or as "superposition" in the mathematical literature of stochastic processes.

As it was shown in many research works, statistical properties of the stochastic process which forms the network traffic have a significant influence on the network functionality and efficiency. Network performance, as captured by throughput, packet delay and loss rate, degrades gradually with increased variance and LRD of aggregated traffic [32][47]. Contrary to the classical assumptions that aggregated network traffic forms a compound Poisson process which is very smooth, LRD traffic does not smooth out but is bursty over many time scales (self-similar). Consequently, network routers and other middle-boxes servicing such traffic experience larger queuing delay and response time [48]. Generally, statistical properties of network traffic fluctuations are a topic of interest as they allow for better understanding of the complex structure and dynamics as well as performance evaluation of adaptive video systems. In practice, this knowledge is usually applied for a design of systems which are able to guarantee the suitable quality of service (QoS) for users [16][29][49].

The amount of multiplexed traffic sent within a certain time period is represented in our work as a counting process. From a mathematical point of view, let N_t be a stochastic process whose values represent a cumulative number of bytes sent until time t. Assuming that $N_t>0$ and $N_t \in \mathbf{I}$, we are interested in the relation

$$C_t = \frac{N_{t+\Delta t} - N_t}{\Delta t}, \tag{3}$$

which is to be interpreted as a number of bytes sent within a Δt period.

6.1 Variability

The variability of time series is popularly measured by the coefficient of variation (CV), which is defined as

$$CV = \frac{\mathrm{Std}(C_t)}{\mathrm{E}(C_t)}, \tag{4}$$

where Std() denotes the standard deviation and E() denotes the mean of the examined time series C_t. Thus, (4) shows the extent of the variability in relation to the mean of time series.

6.2 Long-memory processes

There are several ways of characterising long-memory processes. A widespread definition is in terms of the autocorrelation function $\gamma(k)$. We define a process as long-memory if in the limit $k \to \infty$

$$\gamma(k) \sim k^{-\alpha} L(k), \tag{5}$$

where $0<\alpha<1$ and $L(k)$ is a slowly varying function at infinity. The degree of long-memory is given by the exponent α; the smaller α, the longer the memory. By contrast, one speaks of short range dependent process if the autocorrelation function decreases at a geometric rate and $\alpha>1$.

Long-memory is also discussed in terms of the Hurst exponent H, which is simply related to α from (5). For a stochastic process, $H=1-\alpha/2$ or $\alpha=2-2H$. When $H \in (.5,1]$ the process is positively correlated which implies that it is persistent and is characterised by long-memory effects on all time scales, i.e. the realisation of the process has been up or down in the last period then the chances are that it will continue to be up or down, respectively, in the next period. On the other hand, when $H \in [0,0.5)$, we have long-term anti-persistence what means that whenever the realisation of the process has been up in the last period, it is more likely that it will be down in the next period. With increasing H, the persistence increases, while with decreasing H, the anti-persistence increases, see (5). When $H=1/2$, and the autocorrelation function decays faster than k^{-1}, then the process has no memory. Examples of the compound Poisson and LRD processes are presented in **Fig. 8**.

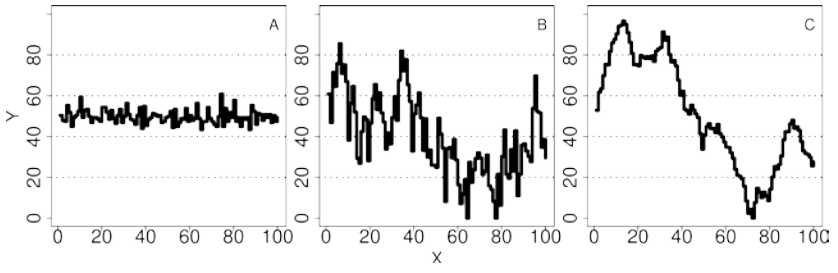


Fig. 8. Example of stochastic processes: a) compound Poisson, b) LRD with $H<1/2$, c) LRD with $H>1/2$. Mean set to 50.

In this paper, to the estimate Hurst exponent, we employ well-known R/S technique. There is a freely available code for the R/S algorithm implemented, among others, in the Rmetrics software [50], which is a part of the Cran R environment [46].

6.3 Results

A visual assessment indicates that both captured traces defined in (3) and presented in **Fig. 9** have an irregular structure. However, in the case of the fixed values of the algorithm parameters, see **Fig. 9A**, the traffic remains in a broader range compared to the traffic generated by the algorithms with random parameters, see **Fig. 9B**. Moreover, the traffic presented in **Fig. 9A** forms more visible trends and frequency of its oscillations is lower compared to the traffic depicted in **Fig. 9B**.

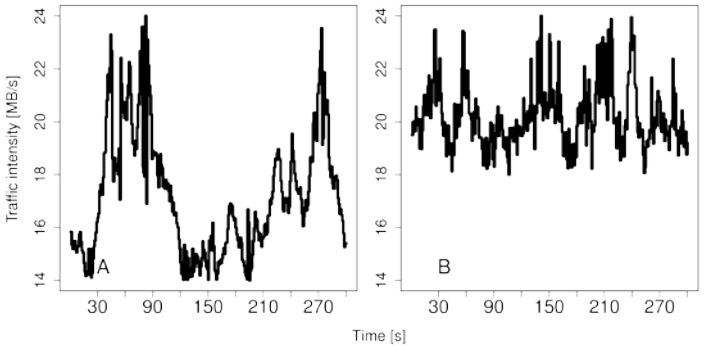
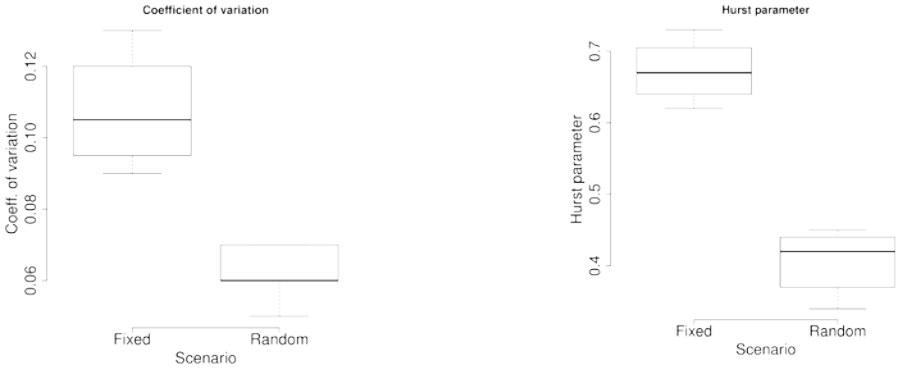


Fig. 9. Intensity of aggregated traffic generated by the adaptive video algorithm with parameters set to a) fixed, b) random values.



(a) Coefficient of variation (b) Hurst parameter

Fig. 10. Variability measures for aggregated traffic generated by algorithms with fixed values and random values of some of the parameters as listed in **Table 1**.

In order to obtain more objective measures, we computed the coefficient of variation **(4)** and the Hurst parameter **(5)**. Randomisation of the parameters values listed in Table 1 significantly reduces the coefficient of variation, from little above 0.10 to about 0.06, see **Fig. 10a**. When it comes to the assessment of the LRD, the values of the Hurst parameter drop from about 0.66 to about 0.42, as presented in **Fig. 10b**. Thus, the randomisation not only decreases Hurst parameter, but also changes the traffic characteristics from persistent to anti-persistent, as defined in **section 6.2**.

In a consequence, the randomisation changes important parameters of the traffic characteristics. It reduces traffic amplitude, variability and persistence, what, from the traffic engineering point of view, usually has a positive impact on network infrastructure. Smoother traffic reduces queues length in buffers of network middle-boxes and, consequently, reduces packet

delays and probability of packets loss.

7. Impact on quality of video

The quality of experience (QoE) is defined as the overall acceptability of an application or service quality perceived by the end-user. The QoE, based on popular methods reflecting human perception, is a subjective assessment of multimedia quality. A user is usually not interested in performance measures such as packet loss probability or received throughput, but mainly in the current quality of the received content. However, the quality assessment is time-consuming and cannot be done in real time; therefore, we concentrate on these parameters which we believe significantly impact the QoE. In this respect, we assess efficiency and stability of the play-out algorithms, which were, among others, employed also in [13].

The efficiency measures how effectively the algorithm utilises available network resources by computing a value of the following formula

$$\text{efficiency} = \left(\sum_{i}^{M} \frac{\sum_{j}^{N} (q_{ij}/\tilde{q_{ij}})}{N}\right)/M. \tag{6}$$

Eq. **(6)** computes the relation between the quality level q_{ij} of the chunk to the theoretical quality level $\tilde{q}_{ij}$ which is possible to achieve for the *j*-th chunk in given network conditions. The theoretical quality level is computed as $\max\{q: q \le n_x\}$. The formula is computed and averaged for every player *i* from M players connected to the server. The minimum value of the formula is zero if the play-out of the video stalls, although, the network conditions allow for at least $Q_{\min}$ quality. The value of the formula can reach one if for every chunk $q_{ij} = \tilde{q}_{ij}$. The periods during which $n_x < Q_{\min}$ (see **Algoritm 1** for the symbols description) result in $\tilde{q}_i = 0$ and, therefore, are excluded from the summation in **(6)**. To avoid unnecessary penalty, we also did not take into account in the summation the cases when $q_{ij} = 0$ and the player buffer level is at least half filled. In such cases, there may be no need to download data.

The play-out algorithm may try to maximise the value of **(6)** by adjusting the play-out quality to given network conditions as frequently as it is possible. Such behaviour will result in rapid oscillations of video quality, what will be negatively perceived by users [51][52]. For this reason, we introduce the second measurement, which sums the quality switches of every player *i* and then counts their average number for *M* players connected to the server. We compute the formula as follows

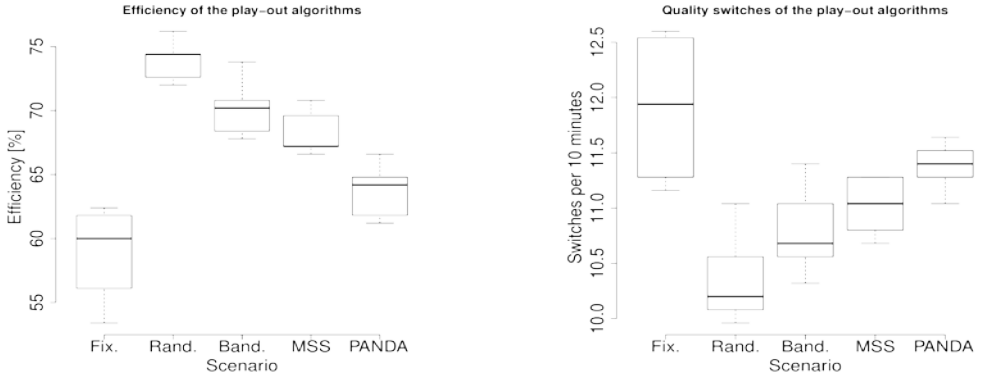
$$\text{switches} = \frac{\sum_{i}^{M} \sum_{j}^{N-1} |q_{ij+1} - q_{ij}|}{M}. \tag{7}$$

Comparing the perceived video quality, we consider five scenarios. In the base scenario, denoted in **Fig. 11a** as *Fix.*, all the parameters of the play-out algorithms have fixed values. In the second scenario, denoted as *Rand.* the selected parameters listed in Table 1 have random values. In the remaining three scenarios, denoted as *Band.*, *MSS* and *PANDA*, only parameters of one selected algorithms are randomised and the other two algorithms operate with fixed values of their parameters. The general set-up of the experiment is as in the previous investigations and as in the description given in **section 4.**

The examination shows that the assignment of random values to the parameters of the

play-out algorithms not only decreases dependency among traffic flows and variability of the aggregated traffic but also improves efficiency and stability of video play-out, see **Fig. 11b**. When the randomisation is applied to all play-out strategies, the efficiency of the streaming measured by **(6)** improves in average about 25%. When the randomisation is applied only to selected algorithms, the improvement is lower but still significant. It ranges from about 20% for pure bandwidth-based strategy presented in **Algorithm** 1 to about 8% for *PANDA* approach. The randomisation of parameter values of the simplest bandwidth-based approach provides the most visible gain in efficiency. While the randomisation is applied to more advanced strategies like *MSS* and *PANDA* especially, the improvement is lower.

Generally, the efficiency measure **(6)** has much better improvement compared to the stability measure **(7)**. When applying randomisation to the parameters of all three strategies, the average number of quality switches drops about 10%. When the randomisation is applied selectively, the most visible improvement achieves the bandwidth-based strategy. The gain for the most elaborate *PANDA* approach is marginal. However, it is the good news: the improvement in throughput utilisation is a result of the employment of the randomisation of algorithms parameters and does not come at the cost of a decrease of play-out stability. Indirectly, the quality improvement may also be a result of smoother and more manageable aggregated traffic produced by the video system.



(a) efficiency of the play-out (b) stability of the play-out

Fig. 11. Performance of a video system using adaptive play-out algorithms.

8. Conclusions

Our investigation shows that the traffic generated by clients employing the same or similar play-out strategies is positively correlated and synchronised, whereas traffic originated from different play-out strategies shows negative or no correlations. However, when some of the parameters of the play-out strategies are randomised, the correlation and synchronisation diminish what has a positive impact on the video quality perceived by end users. Our research shows that non-correlated traffic flows generated by play-out strategies improve efficiency and stability of streamed adaptive video.

In the case of correlated streams, the worse video efficiency and quality may have its roots in the mentioned in **section 3** inaccurate throughput estimation while a video player competes against the other player. Furthermore, the correlated streams increase traffic variability, what, as stated is many works, may result in larger queueing delays and response time. Finally, these conclusions are supported by other research which shows that traffic correlation originating from multiple connections are responsible for packet losses and decrease of network

throughput.

The presented results open new and interesting questions. Firstly, currently, it is unknown whether randomly assigned values to the parameters of adaptive algorithms provide the best possible efficiency and stability of video play-out. More probably, there exists an optimal solution to this problem in which the parameters should acquire values according to some mathematical formula. Moreover, this formula may depend on particular play-out strategy and on other parameters, for example on network configuration or on an amount of video players.

Secondly, to perform diversification, one needs to find a robust solution which is able to assign different parameters for the active play-out algorithms. In one possible scenario, a web server providing video content can participate in this operation, however, this requires feedback from the players.

Finally, the presented approach does not stay in opposition to other proposed solutions in the literature. In contrary, it can be applied to any advanced play-out algorithms as well as it can be employed simultaneously with server-side ideas. However, more analyses are needed to examine the performance of such hybrid solutions.

References

[1] YouTube, "YouTube Statistics". Article (CrossRef Link).

[2] Cisco visual networking index: forecast and methodology, 2014-2019. Article (CrossRef Link).

[3] I. Sodagar, "The mpeg-dash standard for multimedia streaming over the internet," *Multimed. IEEE*, vol. 18, no. 4, pp. 62–67, 2011. Article (CrossRef Link).

[4] T. Kupka, P. al Halvorsen, and C. Griwodz, "An evaluation of live adaptive HTTP segment streaming request strategies," in *Proc. of Local Computer Networks (LCN), 2011 IEEE 36th Conference on*, pp. 604–612, 2011. Article (CrossRef Link).

[5] S. Sen, J. L. Rexford, J. K. Dey, J. F. Kurose, and D. F. Towsley, "Online smoothing of variable-bit-rate streaming video," *Multimed. IEEE Trans. On*, vol. 2, no. 1, pp. 37–48, 2000. Article (CrossRef Link).

[6] A. Zambelli, "IIS smooth streaming technical overview," *Microsoft* Article (CrossRef Link).

[7] T. Stockhammer, "Dynamic adaptive streaming over HTTP–: standards and design principles," in *Proc. of the 2nd annual ACM conference on Multimedia systems*, pp. 133–144, 2011. Article (CrossRef Link).

[8] Apple, "HTTP Live Streaming Resources - Apple Developer" Article (CrossRef Link).

[9] Microsoft Smooth Streaming. Article (CrossRef Link).

[10] Adobe HTTP Dynamic Streaming. Article (CrossRef Link).

[11] C. Müller and C. Timmerer, "A VLC media player plugin enabling dynamic adaptive streaming over HTTP," in *Proc. of the 19th ACM international conference on Multimedia*, pp. 723–726, 2011. Article (CrossRef Link).

[12] J. Famaey, S. Latré, N. Bouten, W. Van de Meerssche, B. De Vleeschauwer, W. Van Leekwijck, and F. De Turck, "On the merits of SVC-based HTTP adaptive streaming," in *Proc. of Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pp. 419–426, 2013.

[13] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran, "Probe and adapt: Rate adaptation for http video streaming at scale," *Sel. Areas Commun. IEEE J. On*, vol. 32, no. 4, pp. 719–733, 2014. Article (CrossRef Link).

[14] S. Akhshabi, L. Anantakrishnan, C. Dovrolis, and A. C. Begen, "What Happens When HTTP Adaptive Streaming Players Compete for Bandwidth?," in *Proc. of NOSSDAV*, 2012. Article (CrossRef Link).

[15]T. Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari, "Confused, timid, and unstable: picking a video streaming rate is hard," in *Proc. of the 2012 ACM conference on Internet measurement conference*, pp. 225–238, 2012. Article (CrossRef Link).

[16] X. K. Zou, J. Erman, V. Gopalakrishnan, E. Halepovic, R. Jana, X. Jin, J. Rexford, and R. K. Sinha, "Can Accurate Predictions Improve Video Streaming in Cellular Networks?," in *Proc. of HotMobile*, 2015. Article (CrossRef Link).

[17] T.-Y. Huang, R. Johari, and N. McKeown, "Downton abbey without the hiccups: Buffer-based rate adaptation for http video streaming," in *Proc. of the 2013 ACM SIGCOMM workshop on Future human-centric multimedia networking*, pp. 9–14, 2013. Article (CrossRef Link).

[18] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "Using the Buffer to Avoid Rebuffers: Evidence from a Large Video Streaming Service," *ArXiv Prepr. ArXiv14012209*, 2014. Article (CrossRef Link).

[19] F. Wamser, D. Hock, M. Seufert, B. Staehle, R. Pries, and P. Tran-Gia, "Using buffered playtime for QoE-oriented resource management of YouTube video streaming," *Trans. Emerg. Telecommun. Technol.*, vol. 24, no. 3, pp. 288–302, 2013. Article (CrossRef Link).

[20] J. Park and K. Chung, "Rate adaptation scheme for HTTP-based streaming to achieve fairness with competing TCP traffic," in *Proc. of Information Networking (ICOIN), 2015 International Conference on*, pp. 222–226, 2015. Article (CrossRef Link).

[21] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," in *Proc. of the 8th international conference on Emerging networking experiments and technologies*, pp. 97–108, 2012. Article (CrossRef Link).

[22] G. Cofano, L. De Cicco, and S. Mascolo, "A control architecture for massive adaptive video streaming delivery," in *Proc. of the 2014 Workshop on Design, Quality and Deployment of Adaptive Video Streaming*, pp. 7–12, 2014. Article (CrossRef Link).

[23] R. Houdaille and S. Gouache, "Shaping http adaptive streams for a better user experience," in *Proc. of the 3rd Multimedia Systems Conference*, pp. 1–9, 2012. Article (CrossRef Link).

[24] S. Akhshabi et all, "Server-based traffic shaping for stabilizing oscillating adaptive streaming players," in *Proc. of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, pp. 19–24, 2013. Article (CrossRef Link).

[25]X. Liu and A. Men, "QoE-aware Traffic Shaping for HTTP Adaptive Streaming," *Int. J. Multimed. Ubiquitous Eng.*, vol. 9, no. 2, 2014. Article (CrossRef Link).

[26] K. Satoda, H. Yoshida, H. Ito, and K. Ozawa, "Adaptive video pacing method based on the prediction of stochastic TCP throughput," in *Proc. of Global Communications Conference (GLOBECOM), 2012 IEEE*, pp. 1944–1950, 2012. Article (CrossRef Link).

[27] Y.-T. Yu and S.-R. Tong, "Adaptive Transmission Control Protocol-trunking flow control mechanism for supporting proxy-assisted video on demand system," *Int. J. Commun. Syst.*, vol. 25, no. 10, pp. 1363–1380, 2012. Article (CrossRef Link).

[28] J.-S. Leu and S.-F. Chen, "TRASS: A transmission rate-adapted streaming server in a wireless environment," *Int. J. Commun. Syst.*, vol. 24, no. 7, pp. 852–871, 2011. Article (CrossRef Link).

[29] N. Bouten, R. de O. Schmidt, J. Famaey, S. Latré, A. Pras, and F. De Turck, "QoE-Driven In-Network Optimization for Adaptive Video Streaming Based on Packet Sampling Measurements," *Comput. Netw.*, 2015. Article (CrossRef Link).

[30] B. J. Villa and P. E. Heegaard, "Detecting period and burst durations in video streaming by means of active probing," in *Proc. of International Journal of Computer and Communication Engineering*, vol. 2, pp. 460–467, 2013. Article (CrossRef Link).

[31] A. Rao, Y. S. Lim, C. Barakat, A. Legout, D. Towsley, and W. Dabbous, "Network Characteristics of Video Streaming Traffic," in *Proc. of CoNEXT*, Japan, 2011. Article (CrossRef Link).

[32] K. Oida, "Propagation of Low Variability in Video Traffic," *J. Netw.*, vol. 10, no. 8, pp. 448–461, 2015. Article (CrossRef Link).

[33] S. Hemminger, "Network emulation with NetEm," in *Proc. of Linux Conf Au*, pp. 18–23, 2005.

[34] The Apache Software Foundation, *Apache Web Server*, Article (CrossRef Link).

[35] G. Hooghiemstra and P. Van Mieghem, "Delay distributions on fixed internet paths," Delft University of Technology, 2001.

[36] R. C. L. Gámez, P. Martí, M. Velasco, and J. M. Fuertes, "Wireless network delay estimation for time-sensitive applications," *Autom Control Dept Tech. Univ Catalonia Catalonia Spain Tech Rep ESAII RR-06-12*, 2006.

[37] A. Botta, A. Dainotti, and A. Pescapè, "A tool for the generation of realistic network workload for emerging networking scenarios," *Comput. Netw.*, vol. 56, no. 15, pp. 3531–3547, 2012.
Article (CrossRef Link).

[38] J. Yao, S. S. Kanhere, I. Hossain, and M. Hassan, "Empirical evaluation of HTTP adaptive streaming under vehicular mobility," in *Proc. of International Conference on Research in Networking*, pp. 92–105, 2011. Article (CrossRef Link).

[39] S. Lederer, C. Müller, and C. Timmerer, "Dynamic adaptive streaming over HTTP dataset," in *Proc. of the 3rd Multimedia Systems Conference*, pp. 89–94, 2012. Article (CrossRef Link).

[40] V. Jacobson, C. Leres, and S. McCanne, "TCPDUMP public repository," *www.tcpdump.org*.

[41] N. Blagus, L. Šubelj, and M. Bajec, "Self-similar scaling of density in complex real-world networks," *Phys. Stat. Mech. Its Appl.*, vol. 391, no. 8, pp. 2794–2802, 2012.
Article (CrossRef Link).

[42] J. Kwapień and S. Drożdż, "Physical approach to complex systems," *Phys. Rep.*, pp. 116–225, 2012. Article (CrossRef Link).

[43] L. Qiu, Y. Zhang, and S. Keshav, "Understanding the performance of many TCP flows," *Comput. Netw.*, vol. 37, no. 3, pp. 277–306, 2001. Article (CrossRef Link).

[44] R. N. Mantegna, "Hierarchical structure in financial markets," *Eur. Phys. J. B*, vol. 11, no. 1, pp. 193–197, Sep. 1999. Article (CrossRef Link).

[45] R. C. Prim, "Shortest Connection Networks And Some Generalizations," *Bell Syst. Tech. J.*, vol. 36, no. 6, pp. 1389–1401, 1957. Article (CrossRef Link).

[46] R Core Team, *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing, 2015.

[47] K. Park, G. Kim, and M. E. Crovella, "Effect of traffic self-similarity on network performance," in *Proc. of Voice, Video, and Data Communications*, pp. 296–310,1997. Article (CrossRef Link).

[48] J. Liebeherr, A. Burchard, and F. Ciucu, "Delay bounds in communication networks with heavy-tailed and self-similar traffic," *Inf. Theory IEEE Trans. On*, vol. 58, no. 2, pp. 1010–1024, 2012. Article (CrossRef Link).

[49] K. Miller, A.-K. Al-Tamimi, and A. Wolisz, "Low-Delay Adaptive Video Streaming Based on Short-Term TCP Throughput Prediction," *ArXiv Prepr. ArXiv150302955*, 2015.

[50] Diethelm Wuertz and et all, *Rmetrics*. 2015. Article (CrossRef Link).

[51] M. Zink, O. Künzel, J. Schmitt, and R. Steinmetz, "Subjective impression of variations in layer encoded videos," in *Proc. of IWQoS 2003*, Springer, pp. 137–154, 2003. Article (CrossRef Link).

[52] P. Ni, A. Eichhorn, C. Griwodz, and P. Halvorsen, "Fine-grained scalable streaming from coarse-grained videos," in *Proc. of the 18th international workshop on Network and operating systems support for digital audio and video*, pp. 103–108, 2009. Article (CrossRef Link).

Arkadiusz Biernacki received his B.Eng. degree in Mathematics and the M.Sc. and Ph.D degrees in Computer Science from the Silesian University of Technology, Poland, in 2000, 2002 and 2007 respectively. From 2007 he is an Assistant Professor at the Silesian University of Technology in Poland. His research interests focus on network traffic modelling, computer system simulations and multimedia performance.