Traffic prediction methods for quality improvement of adaptive video

Arkadiusz Biernacki¹

Received: 24 May 2017 / Accepted: 11 November 2017 / Published online: 25 November 2017
© The Author(s) 2017. This article is an open access publication

Abstract

During the past years, adaptive video based on HTTP has become very popular. Streaming of the adaptive video relies heavily on an estimation of end-to-end network throughput, which can be challenging especially in mobile networks, where the capacity highly fluctuates. In this work, we propose to predict the network throughput using its past measurements. As the analysis shows, the network throughput forms a long range-dependent process; thus, for the throughput prediction, we apply a fractional ARIMA process and artificial neural networks. Our approach does not require any modifications to the network infrastructure or the TCP stack. The predictions are performed for data traces obtained from measurements of throughput of a real mobile network. As the experiment shows, the obtained traffic models are able to enhance the performance of an adaptive streaming algorithm. Compared to the throughput predictors employed in contemporary systems dedicated to adaptive video streaming, the proposed technique obtains better results when taking into account effectiveness of network capacity utilisation and stability of video play-out.

Keywords Traffic prediction · FARIMA processes · Artificial neural networks · Stochastic process · Adaptive video

1 Introduction

Streamed multimedia are susceptible to highly variable packet delays, losses and fluctuating network capacity. Thus, streaming high-quality video to users, especially those located in mobile networks, remains a challenge. Modern mobile networks offer fairly high maximum capacity; however, due to the nature of wireless transmission, inherent variability in signal strength, interference, noise, and user mobility, the capacity usually has a high amplitude of variability. To cope with unstable network environments, content providers split video into segments and encode every segment with different bit rate. With this approach, it is possible to switch the media bit rate (and hence the quality) after each video segment is downloaded and adapt it to the current network conditions [1], see Fig. 1. Thus, users' software is more flexible when a network environment is less stable, for example in wireless or mobile networks.

The rate-adaptation algorithm, deciding which segment should be requested 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 or the dynamics of the available throughput. Furthermore, the client has also to control the filling level of its local buffer to avoid underflows resulting in playback interruptions. The rate adaptation algorithm might work fairly well for the case when the player does not share a connection with other flows, network resources are stable and do not fluctuate. Since network throughput is typically measured using an average of recent throughput, the estimate is typically not the same as the true current throughput. This mismatch results in an undesirable behaviour of the streaming algorithms which can be both too conservative and too aggressive. Some studies have reported many examples of an inaccurate throughput estimation while a video client competed against another video client, or against other TCP flows [2]. In other studies, it was observed that competing streams were unstable and unfair among each other what led to significant video quality variation over time [3]. Therefore, some research works, e.g., [4–6], propose an algorithm based on measurement of buffer occupancy, while the others try to improve the algorithm by predicting the future throughput [7–9].

Communicated by L. Zhou.

✉ Arkadiusz Biernacki
arkadiusz.biernacki@polsl.pl

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

Fig. 1 Architecture of a video adaptive system based on HTTP

In our work, we try to estimate a characteristic and create a model of the network traffic received by a video player. For this purpose, we compute traffic intensity distribution, its autocorrelation and test it for long range dependence (LRD). Then, we try to fit ARIMA, fractional ARIMA (FARIMA) ((F)ARIMA) processes and artificial neural networks (ANNs) based on multilayer perceptrons (MLP) and modified auto-encoder (RDF) to the obtained statistical characteristics of the network traffic using time intervals of 5 s. Finally, we show that the streaming algorithms, based on the approaches implemented in the Microsoft Smooth Streaming (MSS) video player [10] and in the works [11, 12] obtain better performance using the estimations provided by the models than relying only on averaging of past network measurements.

2 Related works

There are lots of works dedicated to a prediction of general internet traffic, which is composed of many aggregated flows originating from a different kind of applications: multimedia, video and music streaming, web browsing, file downloading or peer-to-peer networking. The current review of the prediction methods and comparison of some of them can be found in [13].

In the case of multimedia applications, the problem lays not in a prediction of intensity of the aggregated traffic but in a prediction of a single traffic flow. Thus, the models of the aggregated Internet traffic are not always directly applicable to describe the intensity of a single flow, especially if it is transmitted through an unstable mobile environment. Consequently, the number of works in this field is more limited and developers employ many elaborate techniques to obtain reliable throughput assessments, for example, [14] introduces a stochastic forecast framework to predict future capacity of a cellular link based on historical statistics; in [15], the authors build a regression tree or use the Gaussian random walk to model the traffic [16].

Most of the contemporary video adaptive algorithms use a simple prediction technique and estimate future throughput as an average of a number of past measurements [11, 17, 18]. In the more advanced approach, video players try to optimise video play-out by employing a range spectrum

of methods from different domains. Nevertheless, there are not many works where adaptation algorithms explicitly apply prediction of network throughput. Some authors observe that prediction alone is not sufficient and can, in fact, lead to degraded QoE, although, they claim that when combined with rate stabilization functions (e.g., buffer control of a video player), prediction outperforms existing algorithms which do not employ predictive methods [7]. Therefore, quite often, developers propose hybrid solutions, where the throughput prediction is only part of them. In [19], the authors propose to deploy in-network quality optimization agents, which should monitor the available throughput using sampling-based measurement techniques and optimize the quality of each client, based on throughput prediction. For the prediction, the authors employed an autoregressive model, a support vector regression and ANNs. In [20], the authors study the effect of prediction errors on the performance of three adaptation approaches, buffer based, rate based, and model predictive control. As a predictor, the authors use a harmonic mean of past throughput measurements.

Authors of other works use prediction to estimate the performance of a streaming system in a near future. In [21], the authors try to predict a re-buffering probability of a video player using an empirical cumulative distribution of throughput traces from an off-line database. The authors of [9] use smoothing techniques and an ARIMA process to develop an adaptation algorithm which, for each adaptation decision, maximizes a QoE-based utility function depending on the probability of playback interruptions, average video quality, and the amount of video quality fluctuations.

ANNs have been widely applied to forecasting of time series including also different types of network traffic. Internet traffic prediction based on MLP network can be found, among others, in [22–24]. Generally, there are no conclusive guidelines regarding the process of network traffic prediction with ANNs. Some authors state that the performance of traffic prediction model can be improved by multiplying the number of layers instead of increasing the number of neurons [25], while other research reveals that more complex algorithms are not necessarily better, and there exists a specific range of operating parameters where predictions are generally more accurate [26]. Nevertheless, so-called deep ANNs have become a popular tool employed in machine intelligence. Different from the structure of traditional ANN with a single hidden layer, deep ANNs have multiple hidden layers what should increase their learning capabilities, among others, in the field of time series forecasting. Some types of deep ANNs are constructed by stacking multiple building blocks, such as auto-encoders [27]. The effectiveness of these auto-encoder ANNs have been demonstrated in temperature prediction [28], weather forecasting [29], prediction of traffic flows [30] or prediction of power consumption in a data centre [31].

The comparison presented in [13] shows that FARIMA processes and ANNs have similar approximation errors. However, the best results are achieved for hybrid solutions which employ both FARIMA and ANNs simultaneously [13, 32, 33].

Sometimes, the network physical layer is used to support prediction, as it may give more accurate information about the status of a network environment. For example prediction made in [8, 34] use information about wireless channel quality indicator (CQI). Furthermore, in [8], the authors found also that the time series which models capacity of a mobile link has LRD characteristics and they propose a model which estimates how long given network capacity is likely to remain consistent. The information about the status of a network physical layer is not always possible to obtain what can be a drawback of the above solutions.

In [35], we proposed to employ (F)ARIMA processes and ANNs to enhance performance of adaptive video services. However, the experiment were done on aggregated network traffic produced in an emulated environment. In our current work, we operated on network traces gathered from a real mobile network. Furthermore, for the prediction of network traces, among others, we employ modified recursive auto-encoder which is a multilayer (deep) ANN.

In contrast to all the mentioned works but [8] and [35], we treat the capacity measurement as a long range-dependent process. Hence, for the capacity modelling, we use, among others, a fractal autoregressive model (FARIMA). Contrary to [8], which also consider a network capacity to be LRD, our approach does not require any modifications to the network infrastructure or the TCP stack, as we solely rely on network measurements performed on the application level.

3 Theoretical background

3.1 Video streaming model

A video stream can be modelled as a set of consecutive data chunks $\{c_i\}$ where $i \in \{1, 2, \dots, N\}$. Every chunk contains L seconds of video encoded with different bit rates. Hence, the total length of the video stream is $N \times L$ seconds. The video player can choose to download the video segment c_i with the bit rate $q_i \in \{Q_{\min}, \dots, Q_{\max}\}$ [bit/s], where $Q_{\min}$ and $Q_{\max}$ are the bounds of the set of all available bit-rate levels. The amount of data in segment c_i is then $L \times q_i$. The higher the bit rate is selected by a video player, the higher is also video quality perceived by a user, what can be formalised using an increasing function $f_q(\cdot) : R \rightarrow R$ which maps the selected bit rate q_i to the video quality $f_q(q_i)$ perceived by a user.

The video segments are downloaded into a player buffer, where they wait for the play-out. At the time t_i , a video player starts to download the video chunk c_i . The downloading time

depends on the selected bit rate q_i as well as an average network throughput r_i . At time t_{i+d} the chunk c_i is completely downloaded and at time t_{i+1} , where $t_{i+d} \leq t_{i+1}$, the video player starts to download the next chunk c_{i+1} . We denote by $b_i \in [0, B_{\text{full}}]$ the length of video, counted in time units, stored in the player buffer at the time t_i . The buffer size B_{full} depends on the particular video player and user's storage limitations. If we denote by r_i [bit/s] the average network throughput at time $\Delta t_i = t_{i+d} - t_i$, then we have:

$$t_{i+d} = t_i + \frac{Lq_i}{r_i},$$

where

$$r_i = \frac{1}{\Delta t_i} \int_{t_i}^{t_{i+d}} r dt. \quad (1)$$

While the video segments are being downloaded and played, the amount of data in the player's buffer changes. After the segment c_i is downloaded, the amount of data increases by L seconds. Meanwhile, the buffer occupancy decreases as the user watches the video. Hence, the evolution of the buffer is described as:

$$b_{i+1} = \max[(b_i - t_{i+1} + t_i), 0] + L. \quad (2)$$

If $b_i < t_{i+1} - t_i$, the buffer becomes empty while the video player is still downloading the chunk c_i what freezes the video.

3.2 Adaptation strategies

When at the time t_{i+1} , a video player selects the bit rate q_{i+1} , it has only knowledge of the past throughput $\{r_t, t \leq t_i\}$, while the future one $\{r_t, t > t_i\}$ is not known. However, the video player may try to use a throughput predictor defined as $\{\hat{r}_t, t > t_i\}$. Furthermore, an inspection of the buffer level may also provide supportive information to the player. Thus, taking into account future throughput estimation and occupancy of the buffer, the player selects the quality of the next segment c_{i+1} :

$$q_{i+1} = f(b_i, \{\hat{r}_t, t > t_i\}).$$

The decision on selection of the video bit rate is based on three exemplary approaches. The first approach employs code of the simplified version of the algorithm implemented in the Microsoft Smooth Streaming (MSS) video player and is extensively described in [10]. The second approach, referred in the literature as *Festive*, is based on the implementation described in [11]. The third solution is described in [12] and we will refer to it as *Tian2016*.

In the first algorithm, the bit rate of the video chunk is selected as follows

$$q_{i+1} = \begin{cases} q_i^+ & b_i > B_{\max} \\ q_i^- & b_i < B_{\min} \\ q_i^- & b_i \in [B_{\min}, B_{\max}] \wedge (q_i > \hat{r}_i) \\ q_i & b_i \in [B_{\min}, B_{\max}] \wedge \hat{r}_i - d < q_i < \hat{r}_i \\ q_i^+ & b_i \in [B_{\min}, B_{\max}] \wedge (q_i < \hat{r}_i - d) \end{cases} \quad \begin{matrix} (3a) \\ (3b) \\ (3c) \\ (3d) \\ (3e) \end{matrix}$$

The notation q_i^+ and q_i^- denotes an increase or a decrease of the video bit-rate q_i by one level respectively. The basic idea of the buffer-based part of the algorithm is to select a video bit rate based on the amount of data that is available in the buffer of a player. Thus, when the buffer reaches $B_{\max}$, the system is allowed to increase the bit rate (3a). Similarly, when the buffer is being drained, the selected quality is reduced if the buffer shrinks below the threshold $B_{\min}$ (3b). When the buffer level is between the thresholds $B_{\min}$ and $B_{\max}$, the quality level is selected on the base of throughput estimation $\hat{r}_i$. If the estimated throughput $\hat{r}_i$ is smaller than the current video bit rate q_i , the quality of video is decreased (3c). When the estimated throughput is higher than the current bit rate, but not sufficiently, the bit rate is not changed (3d). If the estimated bit rate is sufficiently higher than the current bit rate, the video quality is increased (3e). Whether the network capacity is sufficiently higher or not, it depends on the parameter d which 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, which 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 quality levels. The idea of the above described algorithm is illustrated in Fig. 2, and we implemented it in the open-source software presented in [36].

The second approach, *Festive*, similarly to the mentioned *MSS* approach, takes into account network capacity estimates 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 *MSS*, *Festive* uses randomised scheduling and takes into account also fairness among video players.

Fig. 2 Adaptation strategy for adaptive video. The algorithm selects the quality of the video taking into account a level of player buffer and a prediction of network throughput obtained from its past measurements

The third approach, *Tian2016*, tries to avoid video rate fluctuations triggered by short-term TCP throughput variations and estimation errors. Furthermore, it increases video rate smoothly when the available network bandwidth is consistently higher than the current rate and quickly decrease video rate upon the congestion level shift-ups to avoid video playback freezes. The solution is based on the proportional–integral–derivative (PID) controller which is a control loop feedback mechanism. The authors claim that the proposed approach is highly responsive to congestion level shifts and can maintain stable video rate in the face of short-term bandwidth variations which occur in mobile networks.

The presented adaptive strategies are examples of many potential strategies and we use it solely for an evaluation purpose.

Different systems use different segment lengths L which usually range from 2 to 10 s; for example, the *MSS* algorithm uses 2–5 s [10]. The duration of the segments have been discussed briefly in [37]. In our work, we set the duration of the video segment to 5 s. Also following [10], we set the total buffer size B_{full} of the player to 35 s, and the thresholds $B_{\min}$ and $B_{\max}$ to 40 and 80% of the total buffer size, respectively. The rest of the parameters of *Festive* were set as in [11]. For the *Tian2016* approach, the parameters were obtained from [12] from sections V and VI-D.

3.3 Quality measures

As mentioned in Sect. 3.2, a video player selects the chunk quality taking into account such parameters of the system as available network capacity or a level of buffer occupancy. Nonetheless, the decision which quality of the video chunk to choose involves a few potentially conflicting issues. Firstly, a user expects that a video player will provide the highest possible quality within the constraint of network capacity. Secondly, the player should keep the playback as smooth as possible by avoiding frequent bit-rate switches. Thirdly, the video player should prevent its buffer starvation in order to avoid freezing video on a user's screen. Thus, for a performance evaluation, we employed three measures which take into account the above guidelines. The measures were also applied, among others, in [18].

This first measure describes how effectively the algorithm utilises available network capacity by computing the value of the following formula:

$$\text{efficiency} = \frac{\sum_{i=1}^N (q_i / \tilde{q}_i)}{N}, \quad \tilde{q}_i > 0. \quad (4)$$

Equation (4) computes the relation between the bit-rate level q_i of the chunk c_i being downloaded to the theoretical bit-rate level $\tilde{q}_i$ which is possible to achieve for the chunk c_i in given network conditions. For example, for

$q_i \in \{300, 600, 1200\}$ kbit/s, when the network capacity is 500 kbit/s, then $\tilde{q}_i = 300$ kbit/s. If the capacity increases to 700 kbit/s, the theoretical bit-rate level also increases to $\tilde{q}_i = 600$. The minimum value of the formula is $Q_{\min}/Q_{\max}$ if the player downloads the video segments encoded in $Q_{\min}$ bit rate while the network allows to download the video segments encoded in $Q_{\max}$. Analogically, the value of the formula can reach $Q_{\max}/Q_{\min}$ if the player downloads the video segments encoded in $Q_{\max}$ bit rate while the network allows to download the video segments encoded in $Q_{\min}$. The periods during which $r_i < Q_{\min}$ result in $\tilde{q}_i = 0$ and, therefore, are excluded from the summation in (4). The adaptation algorithm may try to maximise the value of (4) 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 [38, 39]. For this reason, we introduce the second measure which sums the bit-rate switches:

$$\text{switches} = \sum_{i=1}^{N-1} |q_{i+1} - q_i|. \quad (5)$$

The last measure takes into account total stalling time experienced by a user. After every chunk c_i is downloaded, the measure counts possible buffer under-runs taking into account the buffer level b_i and the amount of data that should be pulled from the buffer $t_i - t_{i-1}$, see also (2). Thus, we have:

$$\text{buffering} = \sum_{i=1}^N \max(\text{sgn}(t_i - t_{i-1} - b_i), 0). \quad (6)$$

Hence, (6) registers every event when the video player, after downloading the chunk c_i , wants to pull from the buffer more than it is possible.

4 Prediction of network throughput

4.1 The prediction problem

In the common approach to the problem of network throughput prediction (and other time series in general), we have the measurements of previous throughput $\{r_t, r_{t-1}, \dots, r_{t-n}\}$ and we wish to predict the value of r_{t+m} , $m > 0$. For this purpose, we apply the predictor $\hat{r}_{t+m}$ which is based on the observations of the past throughput measurements and may be written as

$$\hat{r}_{t+m} = \phi(r_t, r_{t-1}, \dots, r_{t-n}). \quad (7)$$

The problem is to choose ϕ so that $\hat{r}_{t+m}$ is, in some sense, “closest” to r_{t+m} .

Typically, the video adaptive algorithm uses prediction of network throughput for the nearest future r_{t+1} . The prediction is usually an average of past throughput measurements [18]:

$$\hat{r}_{t+1} = \frac{1}{n} \sum_{k=1}^n r_{t+1-k}. \quad (8)$$

This estimator (8) is applied in the base version of adaptive play-out algorithms [1]. In our work, we try to improve the prediction employing (F)ARIMA processes and ANNs.

We adopted the relative error function from [40] p. 14 and defined the relative prediction error (RPE) for assessing goodness of different parameters of the model. The RPE represents a squared difference between the predictions $\hat{r}_{t+m}$ and the observed values r_{t+m} normalized by the observed value r_{t+m} . The normalisation allows us to assess prediction made for different throughput levels. Mathematically, it can be described as

$$\text{RPE} = \sqrt{\sum_{t=1}^N \left(\frac{r_{t+m} - \hat{r}_{t+m}}{r_{t+m}} \right)^2} / N. \quad (9)$$

The lower is the RPE value, the better the model describes empirical data.

4.2 Long-memory characteristics of network traffic

Modeling of video traffic has been extensively studied in the literature. Many measurement experiments have demonstrated that the video traffic in modern communication networks exhibits LRD and self-similarity [41]. As it was already mentioned, the newest research confirms that also the time series which models LTE downlink capacity has long range dependence [8].

There are several way of characterising long-memory processes. A widespread definition is in terms of the autocorrelation function $\gamma(k)$ [42] page 42. We define a process as long-memory if for $k \rightarrow \infty$

$$\gamma(k) \sim k^{-\alpha} L(k), \quad (10)$$

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 $\alpha > 1$, while $\alpha = 1$ indicates a completely uncorrelated series [42] page 52.

Long memory is also discussed in terms of the Hurst exponent H , which is simply related to α from (10). For a stochastic process, $H = 1 - \alpha/2$ or $\alpha = 2 - 2H$. When $H \in (0.5, 1]$, 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 anti-persistence which 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. when $H = 1/2$, and the autocorrelation function decays faster than k^{-1} , then the process has no long-time memory [42] page 52.

While the Hurst parameter is perfectly well-defined mathematically, it may be a difficult property to measure in real life. Tests for the LRD usually require a considerable amount of data because the measurement should be done at tails of the data distribution, where not so many data are available. Furthermore, different methods of the Hurst parameter estimation often give inconclusive or even contradictory results. The assessment results may be biased by trends, periodicity and corruptions in the data. Therefore, some authors suggested applying a “portfolio” of estimators instead of relying on a single estimator, which could give a misleading assessment caused by properties of the process under investigation [43]. Thus, in this paper, we employ three widespread, well-known techniques, which have been used for some time, to estimate the Hurst exponent: R/S , Aggregated Variance (AV) and Differenced Aggregated Variance (DAV). All the chosen techniques have a freely available code and are implemented, among others, in Rmetrics software [44], which is a part of the Cran R environment [45].

4.3 ARIMA and FARIMA processes

ARIMA processes are considered as a classical and universal modelling tools. They can capture the linear and short range dependencies occurring in time series, but when used for general Internet traffic which exhibits LRD, the ARIMA processes have rather poor performance. Nevertheless, due to their versatility and relative easiness of use, they are extensively applied for modelling and forecasting of network traffic, e.g., [13, 46]. In order to capture LRD of network traffic and obtain better accuracy of a model, some research use FARIMA processes [47, 48]. However, the FARIMA processes are computationally more complex and require more elaborate algorithms for estimation of their parameters.

To define ARIMA and FARIMA processes, we will use the neutral variable x_t , which in general may be considered as an element of time series, and in a particular case, it may represent, for example, network throughput measurements. The time series $\{x_t\}$ is an autoregressive (AR) process of order p , if

$$x_t = \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + \dots + \alpha_p x_{t-p} + w_t, \quad (11)$$

where $\{w_t\}$ is white noise and the α_i are the model parameters.

Applying the backward shift operator

$$\mathbf{B}^n x_t = x_{t-n}, \quad (12)$$

(11) can be expressed as a polynomial of the order p

$$\theta_p(\mathbf{B})x_t = (1 - \alpha_1 \mathbf{B} - \alpha_2 \mathbf{B}^2 - \dots - \alpha_p \mathbf{B}^p)x_t = w_t. \quad (13)$$

A moving average (MA) process of order q is a linear combination of the current white noise term and the q most recent past white noise terms, which is defined by

$$x_t = (1 + \beta_1 \mathbf{B} + \beta_2 \mathbf{B}^2 + \dots + \beta_q \mathbf{B}^q)w_t = \phi_q(\mathbf{B})w_t. \quad (14)$$

When AR (13) and MA (14) terms are added together in a single expression, we obtain a time series $\{x_t\}$ which follows an autoregressive moving average (ARMA) process of order (p, q) , denoted as ARMA(p, q) which has the form

$$x_t = \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + \dots + \alpha_p x_{t-p} + w_t + \beta_1 w_{t-1} + \beta_2 w_{t-2} + \dots + \beta_q w_{t-q}.$$

Using the backward shift operator (12), the ARMA(p, q) process may be presented as

$$\theta_p(\mathbf{B})x_t = \phi_q(\mathbf{B})w_t, \quad (15)$$

where θ_p and ϕ_q are polynomials of orders p and q , respectively.

A series $\{x_t\}$ is integrated of order d , if the d th difference of $\{x_t\}$ is white noise $\{w_t\}$, i.e.

$$(1 - \mathbf{B})^d x_t = w_t. \quad (16)$$

A time series $\{x_t\}$ follows an ARIMA(p, d, q) process if the d th differences of the $\{x_t\}$ series are an ARMA(p, q) process (15) and may be expressed as

$$\theta_p(\mathbf{B})(1 - \mathbf{B})^d x_t = \phi_q(\mathbf{B})w_t. \quad (17)$$

A fractionally differenced ARIMA process $\{x_t\}$ is denoted as FARIMA(p, d, q), and has the form of (17) for some $-\frac{1}{2} < d < \frac{1}{2}$. For the fractionally differenced process, the backward shift operator is defined using the following binomial series expansion

$$(1 - \mathbf{B})^d = \sum_{k=0}^{\infty} \binom{d}{k} (-\mathbf{B})^k. \quad (18)$$

To adapt the above defined ARIMA and FARIMA processes to the problem of throughput prediction, we assume that

$$\hat{r}_{i+1} = x_t.$$

4.4 Artificial neural networks

ANNs have been employed in many disciplines with successes, among them in time series prediction. Similarly to the ARIMA and FARIMA processes, for prediction of time series an ANN uses a sliding window containing n most recent observations, see (7). Nevertheless, unlike linear ARIMA and FARIMA, ANNs can handle also non-linear dependencies of time series [49, 50]. Thus, ANNs have been also widely used for network traffic modelling and prediction [51, 52].

From a number of different architectures for ANNs, for our purposes, we employ multilayer perceptrons (MLP) network with a single hidden layer and modified auto-encoder network (RDF).

In the case of the MLP network, we apply commonly used logistic function

$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

The output of the MLP network is given as

$$x_t = \sum_{i=1}^h w_i^2 \sigma \left(\sum_{j=1}^n w_{ij}^1 x_{t-j} + b_{ij}^1 \right) + b^2,$$

where n is the number of inputs and h is the number of neurons in the hidden layer. The variables $\{w_{ij}^1, w_i^2\}$ denote the weights of the connections between the input and the hidden layer, and the hidden layer and the output layer respectively. The variables $\{b_{ij}^1, b^2\}$ denote the bias terms added to the hidden and output layers respectively as presented in Fig. 3.

Similarly to (F)ARIMA processes, the MLP should minimize the RPE defined in (9). For this purpose, the neural network is said to be trained in order to iteratively decrease the RPE. For this purpose, from the variety of available approaches, we chose the popular back-propagation algorithm [53].

The second type of a network applied in the experiments is a recursive auto-encoder (RAE) with an additional correction

Fig. 3 Multilayer perceptrons with logistic function as an activation function

layer (RDF). An RDF network was introduced in [31] and is based on auto-encoder ANNs (AE). The AE aims at learning representation (encoding) for a set of data, typically for the purpose of dimensionality reduction. In its basic implementation, the AE consists of two layers. The input layer usually has more nodes than the output layer and is used to encode the input data into shorter length codes, as shown in Fig. 4. The output layer is denoted as the code layer. Every node from the input layer is connected to every node in the output layer forming full mesh topology. The encoding is achieved by the activation function g , the connection weights W and bias b of the connections between the two layers. The code is calculated by $Y_t = g(WX_t + b)$ where X_t is input data. The weights W' and bias b' are used to decode X' into its original form by $g'(W'Y_t + b')$, which is the reconstruction of the original data. Thus, the reconstruction error is computed as

$$|X_t - g'(W'Y_t + b')|, \quad (19)$$

which through the AE training should be minimized.

Recursive AE (RAE) proposed in [54] is an extension of the AE. Denoting as X_t all input data $\{x_{t-1}, x_{t-2}, \dots, x_{t-n}\}$ at time slot t , the output data is obtained recursively in the following way: firstly, X_{t-m} and X_{t-m+1} are encoded into the layer E_m , then the layer E_{m-1} and X_{t-m+2} are encoded into the layer E_{m-1} , then the layer E_{m-2} and X_{t-m+3} are encoded into the layer E_{m-2} etc. Finally, the layer E_1 is used to predict x_t . Figure 5 illustrates an example RAE with m parameter set to 3.

According to [31], adding another correction layer can reduce the variance of the RAE output, thus providing more stable prediction results. Thus, to correct the output of the RAE, we have an additional layer (z-NN) which is an AE with input and output layers of dimension equal to m . z-NN takes the data X_{t-m} and differential data $X_{t-m} - X_{t-m+1}, X_{t-m+1} - X_{t-m+2}, \dots, X_{t-1} - X_t$ as the input data and computes $z[0]_t, z[1]_t$, and $z[2]_t$ which are considered the control parameters of the correction layer, see Fig. 6. The adjustment of prediction results of the RAE is supported by the z-NN as the output of RAE is modified and takes form

$$\tilde{x}_t = (x_t - z[0]_t)z[1]_t + z[2]_t.$$

Fig. 4 An auto-encoder consisting of two layers can encode the input data into codes with shorter length

Fig. 5 Recursive auto-encoder with m parameter set to 3

Fig. 6 Recursive auto-encoder with a correction layer z-NN

The training of z-NN aims at minimization of the prediction error of the whole RDF network by adjusting the weight parameters of z-NN.

To train the RAE network, the optimization objective is to minimize both the reconstruction error (19) of the RAE and the RPE (9). After this process, the output x_t of the trained RAE is used as a parameter of the training process of z-NN. The details of this process can be found in [31] in Algorithm 1.

5 Experiments and their results

5.1 Traffic characteristics

The network capacity traces were obtained from the measurements conducted in a mobile wireless HSPA network. Because users of streamed video are more interested in download capabilities, we registered only downloaded data. For this purpose, we implemented software which registered time and amount of downloaded UDP packets. The sender was a workstation in a close proximity of an Internet exchange point. The workstation was connected to the exchange point with a very stable wired connection with

round-trip time of about 3 ms in average and a standard deviation of the round-trip time less than 1 ms. As the receiver, we used a laptop with a wireless connection. The receiver was moving with a different speed and was downloading data in several various places. In the experiments, we sent packets from the workstation at a fixed rate to the laptop which logged their reception. As the transmitted packets had sequence numbers, the laptop was able to compute transmission time of the packets and detect possible loss of the transferred data. When the sender transmission rate exceeded the available capacity of the mobile network, the receiver registered a reduced reception rate. In the experiments, the capacity fluctuation between the sender and the receiver were above all caused by the mobile network; thus, we were able to obtain network traces which quite reasonably approximated the wireless link capacity.

To examine the download capability of the mobile network at time Δt_i , the workstation sent UDP packets at a rate 20% higher than the throughput of the mobile network link registered at time Δt_{i-1} , where $\Delta t_i - \Delta t_{i-1} = 1$ s. After several such tests, we were able to assess that the average network capacity was between 2.8 and 3.7 Mbit/s depending on the receiver location. We gathered five traces, each of them has a duration between 34 and 48 min and their total duration was 202 min. The traces are geographically continuous, i.e., we started to register a new trace in the same location where we ended to register a previous trace. The summary of the traces was presented in Table 1.

The average amount of traffic registered within a certain time period is represented in our work as the network average throughput r_i (1). As stated in Sect. 3.1, the length of

Table 1 Capacity traces used in the experiments

Trace	Activity	Length (min)	Average (Mbit/s)	Variance/mean (Mbit/s)
T1	No movement	42	3.7	2.8
T2	Walking	48	3.6	2.5
T3	Riding a bike	38	3.0	2.5
T4	Riding a bus	34	2.9	2.4
T5	Riding a car	40	2.8	2.2

a video segment is L , and if we assume that the algorithm downloads video segments at the rate $q_i \approx \tilde{q}_i$, the algorithm will measure the network throughput roughly every L seconds. Hence,

$$\Delta T \approx L = 5 \text{ s.} \quad (20)$$

A visual assessment of the longest registered T2 trace indicates that the pattern of the traffic defined in (1) and presented in Fig. 7 has rather an irregular structure regardless of the aggregation scale. At the macro-scale, the traffic remains in a broad range; and there are clearly visible cycles which repeat themselves about every 500 s, as shown in Fig. 7a. At the medium scale, Fig. 7b, the traffic shows an increasing trend which lasts about 1 min and ends with small oscillations. The range of the traffic is quite broad and takes values from nearly zero to more than 4 Mbit/s. The traffic exhibits lots of variability even on the micro-scale, as presented in Fig. 7c.

The empirical investigation of LRD processes must take into account possible non-stationarity in short-time scales, as presented in Fig. 7. During the statistical analysis, it is, therefore, important to try to eliminate or at least minimize

Fig. 7 Fragment of T2 trace representing registered network capacity for ΔT (1) set to **a** 25 s, **b** 5 s and **c** 1 s

such linear dependencies, since it can bias the Hurst parameter. To eliminate the linear dependencies, the autocorrelation and the estimation of the Hurst parameter were obtained for the de-trended throughput R_i (1) computed as

$$R_i = \ln(r_{i+1}/r_i), \quad (21)$$

where ΔT in (1) was set to 1 s.

The significant correlation between the subsequent de-trended intensity values R_i of the exemplary T2 trace in the first 10 s is mostly positive, see Fig. 8a (the first element of the autocorrelation, which is always equal to one, was removed). The positive correlation indicates the persistent nature of the examined traffic traces, what in a consequence means that an increasing trend in the past is likely to be maintained in the future.

The distribution of the traffic intensity r_i of the exemplary traffic trace is negatively skewed—the right tail of the distribution, presented in Fig. 8b, drops abruptly, what is the result of the capacity caps imposed by network infrastructure, while the left tail decays hyperbolically, breaking its slide for the parameters slightly above zero.

As presented in Fig. 8c, the Hurst parameter for the all gathered traces oscillates between 0.55 and 0.70, what confirms the persistent nature of the traffic revealed previously by the visual inspections and the autocorrelation analysis.

5.2 Traffic predictions with ARIMA and FARIMA processes

We applied the Box–Jenkins procedure in order to adjust the ARIMA and FARIMA processes to the obtained traffic traces. The details of the procedure is provided in the

Fig. 8 Estimated **a** autocorrelation with 95% confidence bands, **b** density for an exemplary network trace and **c** Hurst parameter for the all examined network traces

literature, for example in [55] in section 2.3. The adjustment procedure consists of several steps: an estimation of the model order, an estimation of the model coefficients, a diagnostic check of the obtained model and its prediction accuracy. From every of five examined traces we extracted three 15 min length sub-traces which might overlap. The models were trained on the first half of each sub-trace and tested against the RPE (9) on the second half of the sub-trace.

To determine the order of the models, we employed Akaike's Information Criterion (AIC), presented, among others, in [56] in section 8.6, and defined as

$$\text{AIC} = -2 \log(L) + 2(p + q + k + 1), \quad (22)$$

where L is the maximum value of the likelihood function for the model and k is the number of estimated parameters in the model. Rather than checking all their possible combinations of parameters p and q , the algorithm uses a gradient descent to minimize (22). To avoid overfitting of the models, we set a limit on the number of parameters $p, q \leq 3$. From the set of candidate models, we selected one ARIMA (A2) and two FARIMA (F1 and F2) models, as presented in Table 2. The ARIMA(A1), plays a role of a reference model being an implementation of the simple throughput predictor specified in (8) and employed, among others, in [18].

For the estimation of ARIMA parameters, we used the routines implemented in the *arima* library, which is a part of the standard CRAN R distribution [45]. While estimating the FARIMA parameters, we had to apply additional calculation, as the *arima* routines operate only on integer values of the differencing parameter d . Firstly, using the relationship between the differencing parameter d and the long-memory parameter α from (10), we estimated the value of d , which according to [57] section 8.2, is

$$d = \frac{1 - \alpha}{2} \rightarrow d = H - \frac{1}{2}.$$

Table 2 Exemplary parameters of the examined models for T3 trace

Model	Short	Parameters		
		α	d	β
ARIMA(4, 0, 0)	A1	$\alpha_i = 0.25$	0	0
ARIMA(2, d , 1)	A2	$\alpha_1 = 0.35$ $\alpha_2 = 0.14$	1	-0.79
FARIMA(2, d , 1)	F1	$\alpha_1 = -0.59$ $\alpha_2 = 0.17$	1.15	-0.64
FARIMA(3, d , 1)	F2	$\alpha_1 = -0.42$ $\alpha_2 = 0.12$ $\alpha_3 = 0.07$	1.15	-0.48

Except of the A1 model, the parameters of other models vary in different experiments

From the analysis presented in Fig. 8, the median of Hurst parameter is about 0.65 what implies that the differencing parameter $d = 0.15$. After the estimation of the above parameter, the time series was fractionally differenced. The calculation was based on the binomial expansion of $(1 - B)^d$, given by (18), can be presented as

$$(1 - B)^d = 1 - dB + \frac{d(d-1)}{2!}B^2 - \frac{d(d-1)(d-2)}{3!}B^3 + \dots,$$

and curtailed at some suitably large lag L , which we set to 40 following [57] section 8.2. Thus, for $d = 0.15$ we get

$$y_t = x_t - 0.15x_{t-1} - 0.06375x_{t-2} - \dots - 0.0019425x_{t-40}.$$

To the fractionally differenced time series $\{y_t\}$, we applied the *arima* library in order to get estimates of the model order and the values of the p and q parameters. The estimated exemplary parameters for the models are presented in Table 2.

5.3 Traffic prediction with neural networks

The important factor of an ANN are the number of input variables, the number of hidden layers and number of nodes for each layer. In Sect. 4.4 we have already defined the basic architecture of the MLP and RDF networks. The former consist of three layers, which include one hidden layer; the latter has five layers, which include three hidden layers. Additionally, there is the correction layer. In the rest of this section we focus on the number of input and hidden nodes.

To estimate the number of input neurons, it is assumed that the analysed data is produced by an unknown M -dimensional dynamic system. However, the embedding theorem [58] allows us to identify a simpler N , $N < M$ dimensional system which is considered as an equivalent to the original system. Furthermore, the data are obtained from the past values $\{y_t\}$ with a sampling rate τ , $\tau \in \mathbb{N}$. Hence, taking into account (7), we have

$$\hat{x}_{t+m} = \phi(x_t, x_{t-1}, \dots, x_{t-(n-1)}) \equiv \theta(y_t, y_{t-\tau}, \dots, y_{t-(N-1)\tau}), \quad (23)$$

where n denotes the length of an input vector X_t of an ANN. The equivalent system dimension N can be estimated using a classical method called False Nearest Neighbors (FNN) [59]. The main idea behind this method is an analysis how the number of neighbours of a point along a signal trajectory changes with an increasing embedding dimension. If the embedding dimension is too low, many of the neighbors will be false. However, if the embedding dimension is appropriate, the neighbours become real. Therefore, by examining how the number of neighbours change as a function of dimension, the algorithms determine the correct embedding dimension.

In the case of the MLP, the number of nodes in the input layer is equal to the length of the input vector X_t . In the case of the RDF, as we set $m = 3$ and every auto-encoder have three input neurons (following [54]), the total number of input neurons is 12, see Fig. 5. Taking into account the z-NN, which has three input and three output layers, the total number of input neurons increases to 15. Also the number of neurons in the hidden layer is increased by three in order to capture the output of z-NN.

The sampling rate τ was estimated by Mutual Information (MI) method [60]. This procedure, which can be an equivalent of the correlation function but in a non-linear environment, uses the MI function. The first local minimum of this function is selected as the sampling rate τ .

Table 3 presents the summary of the most important networks attributes. The sampling rate $\tau = 1$ denotes that the network throughput will be probed accordingly to (20) every 5 s. We performed the computations in *CRAN R* environment employing *nnet* package [61].

Table 3 Basic parameters of the ANNs applied in the experiments

Model	Short	Parameters			τ
		Layers	Neurons		
Multi-layer perceptron	MLP	3	(6, 9, 1)		1
Recursive auto-encoder with a correction	RDF	5 + 1	(12 + 3, 3, 3, 3, 1 + 3, 1)		1

Fig. 9 Distribution of errors for the one-step prediction for **a** A1, **b** A2, **c** F1, **d** F2, **e** MLP and **f** RDF models. Estimated on T3 trace

5.4 Models validation

To check if the prediction errors are normally distributed with mean zero, we plotted a histogram of the forecast errors for each examined model, with an overlaid normal curve that has a mean zero and the same standard deviation as the distribution of prediction errors, see Fig. 9. The plot shows that the distribution is roughly centred on zero, and is more or less normally distributed. Although, in the case of the A2 and F2 models, the tails of the distribution are little heavier compared to the normal curve. However, this aberration is relatively small, and we can assume that the prediction errors are normally distributed with mean zero. We may also observe that variability of the prediction made by F2, MLP and RDF models is smaller compared to the simpler ARIMA processes.

Except the above visual assessment, the distribution of the one-step prediction errors was verified using the Kolmogorov–Smirnov (KS) test. The KS test is a nonparametric statistics that can be used to compare a sample $F_n(x)$ with a reference probability distribution $F(x)$. In our case, the KS statistic quantifies a distance between the cumulative distribution of the one-step prediction errors and the normal distribution. Our null hypothesis is that the one-step prediction errors have the normal distribution.

The KS statistic for a given cumulative distribution function $F(x)$ is

$$D_n = \sup_x |F_n(x) - F(x)|, \quad (24)$$

where $\sup_x$ is the supremum of the set of distances. The null hypothesis is rejected at level α if

$$D_{n,n'} > c(\alpha) \sqrt{\frac{n+n'}{nn'}}, \quad (25)$$

where n and n' are the sizes of first and second sample respectively. The value of $c(\alpha)$ for $\alpha = 0.05$ is 1.36 [62] appendix C. We had 90 samples of one-step prediction errors and compared it with the data taken from sampling the normal distribution, what gives $n = n' = 90$, and consequently $c(\alpha) \sqrt{\frac{n+n'}{nn'}} \approx 0.20$. Thus, taking into account the D_n values computed from (24) and associated p values presented in Table 4, the null hypothesis cannot be rejected.

To represent the uncertainty of the throughput estimate, for our predictor $\hat{r}_i$ (7), we applied confidence bands which cover with 80% probability the one-step prediction. In mathematical terms, a point-wise confidence band w_i with coverage probability of 80%, satisfies the following condition separately for each throughput measurement r_i :

$$\Pr(\hat{r}_i - w_i \leq r_i \leq \hat{r}_i + w_i) = 80\%. \quad (26)$$

Table 4 Results for the Kolmogorov–Smirnov test

Model	D_n	p value
A1	0.11	0.38
A2	0.11	0.40
F1	0.12	0.39
F2	0.17	0.36
MLP	0.16	0.46
RDF	0.17	0.43

Figure 10 shows the confidence bands for one-step ahead prediction of a fragment of the capacity trace showed in Fig. 7. From the analysis presented in Fig. 10a, we may observe that in the most cases the values of the predictor based on averaging of the throughput past values is outside the prediction bands provided by the A2 model. The prediction bands for the F1 models, presented in Fig. 10b, and the F2 model, presented in Fig. 10c, are usually more compact compared to the prediction bands of the A2 model. However, there are events, where the FARIMA processes provide less accurate estimations, for example at about 1400th s for the F1 modes, and at about 1270th s for the F2 model. Nevertheless, in most cases, the FARIMA processes may be expected to have a better accuracy compared with the ARIMA and with the simple predictor (8) based on averaging of past measurement. The model based on the MLP delivers quite a good match except for a slip near the end of the presented bandwidth trace as shown in Fig. 10d. The RDF-based prediction handles better abrupt changes; nevertheless, sometimes it loses its accuracy when the bandwidth is relatively stable.

The RPE for the obtained models for the exemplary T3 trace is presented in Fig. 11. Regarding the (F)ARIMA processes, the best fits were obtained for the FARIMA(F2) the FARIMA(F1) based models with the RPE little higher than 0.25. The ARIMA(A2) gained RPE at about 0.33, while the highest RPE had the ARIMA(A1) model at about 0.36. The MLP-based model achieved results similar to F2 model; however, its variance is slightly lower, what may be a result

Fig. 10 Confidence bands for one-step ahead prediction for the **a** A2, **b** F1, **c** F2, **d** MLP and **e** RDF models. Estimated on T3 trace

Fig. 11 RPE of the analysed prediction models for T3 trace

of the mentioned in Sect. 4.4 back-propagation algorithm. The lowest RPE was generated by the RDF model. The results should not be surprising: the best scores acquired the most elaborate models, the poorest score belongs to the simplest solutions.

5.5 Performance of the play-out algorithms

We conduct the performance study using a trace-driven emulation. The emulation approach allows us to methodologically explore the behaviour of the examined system with different prediction models using the same capacity traces what would be a challenging task if we conducted 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 preserves a high level of accuracy for the obtained results. The captured traffic trace, which was examined in Sect. 5.1, was used as a template for the capacity shaper implemented in the *netem* module, see Fig. 12. In addition to the capacity throttling, the *netem* module also adds a uniformly distributed delay from interval [5 ms, 145 ms] and uniformly distributed packet loss from interval [0.05%, 0.15%] to emulate the instability of the connections which was measured

Fig. 12 Experiment set-up which includes: a web server, a router with the *netem* module and a video player

Fig. 13 A fragment of the throughput trace of a file transmitted through the laboratory set-up with imposed network capacity caps

during the gathering of the capacity traces. Thus, having identical capacity traces, we were able to perform a quite fair and realistic comparison of the different prediction models.

Figure 13 shows a fragment of the throughput trace of a file transmitted through our laboratory set-up with the capacity caps imposed by the capacity shaper. The whole file had 40 MB and was transmitted using HTTP. TCP using the network congestion-avoidance mechanism restricts the achievable network throughput according with the registered earlier capacity trace.

At the web server, presented in Fig. 12, we placed six video files, acquired from [63] and encoded in 300, 600, 1200, 2500 and 4400 kbit/s. These video files were streamed to the VLC media player with the embedded 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 the 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 in Sect. 3.2, we replaced the default logic implemented in the plug-in with the implementation of the simplified version of *MSS* algorithm defined in (3a)–(3e), and two algorithms described in the literature: *FESTIVE* and *Tian2016*. The default throughput predictors in the mentioned algorithms were replaced by the predictors based on the models presented in Table 2. Additionally, we compared the performance of the prediction models with the perfect predictor $\hat{r}_{i+1} = r_{i+1}$, which was denoted as P.

As it could be expected from the analyses presented in Figs. 9, 10 and 11, the traffic models are able to improve the quality of video streaming of the approach based on *MSS* algorithm. In the case of the efficiency assessment defined

Fig. 14 Performance comparison of the prediction models applied to play-out of adaptive video based on *MSS* algorithm

in (4), the A1 model, which estimates future throughput as a simple moving average, achieves the worst score, as presented in Fig. 14a. The difference between the average results obtained by base A1 model and A2 model are of several percents. Nonetheless, the FARIMA-based model achieve much better average results, as the quality of video play-out is about 20% higher for the F1 and F2 models compared to the A1 model. The variability of the results is a little higher for the FARIMA-based models, thus, sometimes there are sporadic situations, where the estimations delivered by the models are inaccurate and worsens the play-out quality. This may happen when, during the play-out, models accuracy gradually deteriorates and their parameters need more frequent recalibration, for example, every minute instead of every seven and half minutes. Compared to the other models, the perfect predictor P significantly increases the efficiency.

The result obtained by the ANNs model based on the MLP is comparable with the FARIMA-based models, while the model based on RDF is slightly better than the other models.

The influence of the predictive models on the video stability, which was defined in (5) is roughly similar, see Fig. 14b. Only a player employing A1 model may experience about 15% more bit-rate switches compared to the other models. Finally, as presented in Fig. 14c, the buffering time measured by (6) is roughly on the same level for all four models and the differences among the models do not exceed 10%. The perfect predictor P does not influence the stability and the buffering time.

Similarly to the approach based on *MSS* algorithm, also in the case of *FESTIVE*, the more complex models increase the play-out efficiency and do not significantly influence stability or buffering time, see Fig. 15. The average efficiency of

Fig. 15 Performance comparison of the prediction models applied to play-out of adaptive video based on *FESTIVE* algorithm

Fig. 16 Performance comparison of the prediction models applied to play-out of adaptive video based on *Tian2016* algorithm

FESTIVE is lower compared to the approach based on *MSS* algorithm. Furthermore, *FESTIVE* approach has higher buffering times. However, the lower efficiency and higher buffering times are compensated by better stability of the play-out.

In the case of *Tian2016* approach, the more advanced predictive model are able to increase the efficiency of the play-out; however, the improvement is less significant compared to *MSS* or *FESTIVE* algorithms, see Fig. 16. Again, the better efficiency does not have negative impacts in the terms of the play-out stability or buffering times.

We can conclude that the more complex models F1, F2 and ANNs are able to increase the efficiency of video play-out and simultaneously this improvement does not come at the cost of a decreased stability or robustness. In the case of older and simpler play-out algorithms like *MSS*, the improvement is more significant compared to the newer and more complex approaches like *Tian2016*. Comparing the results with the perfect predictor, we can see that there is still room for improvement in the prediction accuracy.

6 Conclusions

In this work, we proposed to improve performance of adaptive streaming algorithms employing for this purpose prediction of network throughput. Taking into account the results of the traffic characteristic analysis, the prediction is based on (F)ARIMA processes and ANNs. The ARIMA process is computationally simpler, nevertheless, the FARIMA process fits better the long range-dependent data. As we showed, the prediction based on both models types can improve performance of adaptive play-out algorithms. The ANN based on multilayer perceptrons networks obtains results which are

comparable with the FARIMA process, while the modified auto-encoder, more advanced (deep) ANN, receives the best results from all the models. In the practice, it means that an end user should experience, on average, higher video quality without longer buffering times and without a higher number of bit-rate switches.

There may be doubts, how will the models behave when, e.g., the architecture of the system or the adaptation algorithm change. Of course, there will be need to recompute parameters of the models and update them in certain time intervals. Due to their universality, the models with great probability should handle also traffic generated by systems with different configuration.

There is also a further ground of improvement for the traffic models. Some authors report that the best results are usually obtained by hybrid techniques, for example combination of (F)ARIMA and different types of ANNs [13, 46]. Thus, these more elaborate approaches may give a better adjustment to network data and provide better input for adaptive algorithms.

Furthermore, our approach does not require any modifications of the network infrastructure or the TCP stack. As the bandwidth estimation procedure is usually clearly separated from other modules of a video player, our approach allows for an easy integration with many existing play-out algorithm. The complexity of play-out algorithms does not influence the integration process, nevertheless, the more advanced the play-out algorithm is, the less it gains from bandwidth prediction. The proposed technique can be used directly by developers of the video players which are free to employ their own adaptation strategies. Potentially, they may take into account prediction of network throughput to enhance video play-out.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. Stockhammer, T.: Dynamic adaptive streaming over HTTP—standards and design principles. In: Proceedings of the Second Annual ACM Conference on Multimedia Systems, pp. 133–144 (2011)
2. Huang, T.-Y., Handigol, N., Heller, B., McKeown, N., Johari, R.: Confused, timid, and unstable: picking a video streaming rate is hard. In: Proceedings of the 2012 ACM Conference on Internet Measurement Conference, pp. 225–238 (2012)
3. Houdaille, R., Gouache, S.: Shaping http adaptive streams for a better user experience. In: Proceedings of the 3rd Multimedia Systems Conference, pp. 1–9 (2012)
4. Huang, T.-Y., Johari, R., McKeown, N., Trunnell, M., Watson, M.: Using the buffer to avoid rebuffer: evidence from a large video streaming service. arXiv preprint [arXiv:1401.2209](https://arxiv.org/abs/1401.2209) (2014)
5. Huang, T.-Y., Johari, R., McKeown, N.: Downton abbey without the hiccups: buffer-based rate adaptation for http video streaming. In: Proceedings of the 2013 ACM SIGCOMM Workshop on Future Human-centric Multimedia Networking, pp. 9–14. ACM (2013)
6. Wamser, F., Hock, D., Seufert, M., Staehle, B., Pries, R., Tran-Gia, P.: Using buffered playtime for QoE-oriented resource management of YouTube video streaming. *Trans. Emerg. Telecommun. Technol.* **24**(3), 288–302 (2013)
7. Zou, X.K., Erman, J., Gopalakrishnan, V., Halepovic, E., Jana, R., Jin, X., Rexford, J., Sinha, R.K.: Can Accurate predictions improve video streaming in cellular networks? In: Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications, pp. 57–62. ACM (2015)
8. Xie, X., Zhang, X., Kumar, S., Erran Li, L.: pistream: physical layer informed adaptive video streaming over lte. In: Proceedings of the 21st Annual International Conference on Mobile Computing and Networking, pp. 413–425. ACM (2015)
9. Miller, K., Al-Tamimi A.-K., Wolisz, A.: Low-delay adaptive video streaming based on short-term TCP throughput prediction. arXiv preprint [arXiv:1503.02955](https://arxiv.org/abs/1503.02955) (2015)
10. Famaey, J., Latré, S., Bouten, N., Van de Meerssche, W., De Vleeschauwer, B.: Werner Van Leekwijck, and Filip De Turck. On the merits of SVC-based HTTP adaptive streaming. In: 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), pp. 419–426. IEEE (2013)
11. Jiang, J., Sekar, V., Zhang, H.: 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, pp. 97–108. ACM (2012)
12. Tian, G., Liu, Y.: Towards agile and smooth video adaptation in HTTP adaptive streaming. *IEEE/ACM Trans. Netw.* **24**(4), 2386–2399 (2016)
13. Katris, Christos, Daskalaki, Sophia: Comparing forecasting approaches for Internet traffic. *Expert Syst. Appl.* **21**(42), 8172–8183 (2015)
14. Winstein, K., Sivaraman, A., Balakrishnan, H., et al.: Stochastic forecasts achieve high throughput and low delay over cellular networks. In: NSDI, pp. 459–471 (2013)
15. Xu, Q., Mehrotra, S., Mao, Z., Li, J.: PROTEUS: network performance forecast for real-time, interactive mobile applications. In: Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services, pp. 347–360. ACM (2013)
16. Bui, N., Widmer, J.: Modelling throughput prediction errors as Gaussian random walks. In: The 1st KuVS workshop on anticipatory networks. Stuttgart, Germany (2014)
17. Miller, K., Quacchio, E., Gennari, G., Wolisz, A.: Adaptation algorithm for adaptive streaming over HTTP. In: Packet Video Workshop (PV), 2012 19th International, pp. 173–178 (2012)
18. Li, Z., Zhu, X., Gahm, J., Pan, R., Hao, H., Begen, A.C., Oran, D.: Probe and adapt: rate adaptation for http video streaming at scale. *IEEE J. Sel. Areas Commun.* **32**(4), 719–733 (2014)
19. Bouten, N., Schmidt, R.O., Famaey, J., Latré, S., Pras, A., De Turck, F.: QoE-driven in-network optimization for adaptive video streaming based on packet sampling measurements. *Comput. Netw.* **81**, 96–115 (2015)
20. Yin, X., Jindal, A., Sekar, V., Sinopoli, B.: A control-theoretic approach for dynamic adaptive video streaming over HTTP. In: Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, pp. 325–338. ACM (2015)
21. Liu, Y., Lee, J.Y.B.: On adaptive video streaming with predictable streaming performance. In: Global Communications Conference (GLOBECOM), 2014 IEEE, pp. 1164–1169. IEEE (2014)
22. Cortez, P., Rio, M., Rocha, M., Sousa, P.: Multi-scale Internet traffic forecasting using neural networks and time series methods. *Expert Syst.* **29**(2), 143–155 (2012)
23. Miguel, M.L.F., Penna, M.C., Nievola, J.C., Pellenz, M.E.: New models for long-term Internet traffic forecasting using artificial neural networks and flow based information. In: Network Operations and Management Symposium (NOMS), 2012 IEEE, pp. 1082–1088. IEEE (2012)
24. Gowrishankar, S., Satyanarayana, P.S.: A time series modeling and prediction of wireless network traffic. *Int. J. Interact. Mob. Technol. (iJIM)* **3**(1), 53–62 (2008)
25. Rutka, G.: Neural network models for Internet traffic prediction. *Elektronika ir Elektrotechnika* **68**(4), 55–58 (2015)
26. Liu, Y., Lee, J.Y.B.: An empirical study of throughput prediction in mobile data networks. In: 2015 IEEE Global Communications Conference (GLOBECOM), pp. 1–6. IEEE (2015)
27. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. *Science* **313**(5786), 504–507 (2006)
28. Romeum, P., Zamora-Martínez, F., Botella-Rocamora, P., Pardo, J.: Time-series forecasting of indoor temperature using pre-trained deep neural networks. In: International Conference on Artificial Neural Networks, pp. 451–458. Springer (2013)
29. Liu, J.N.K., Hu, Y., You, J.J., Chan, P.W.: Deep neural network based feature representation for weather forecasting. In: Proceedings on the International Conference on Artificial Intelligence (ICAI), p. 1. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp) (2014)
30. Lv, Y., Duan, Y., Kang, W., Li, Z., Wang, F.-Y.: Traffic flow prediction with big data: a deep learning approach. *IEEE Trans. Intell. Transp. Syst.* **16**(2), 865–873 (2015)
31. Li, Y., Hu, H., Wen, Y., Zhang, J.: Learning-based power prediction for data centre operations via deep neural networks. In: Proceedings of the 5th International Workshop on Energy Efficient Data Centres, p. 6. ACM (2016)
32. Khashei, M., Bijari, M.: A new class of hybrid models for time series forecasting. *Expert Syst. Appl.* **39**(4), 4344–4357 (2012)
33. Khashei, M., Bijari, M., Ardali, G.A.R.: Improvement of autoregressive integrated moving average models using fuzzy logic

- and artificial neural networks (ANNs). *Neurocomputing* **72**(4), 956–967 (2009)
34. Lu, F., Du, H., Jain, A., Voelker, G.M., Snoeren, A.C., Terzis, A.: CQIC: Revisiting cross-layer congestion control for cellular networks. In: *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, pp. 45–50. ACM (2015)
 35. Biernacki, A.: Analysis and modelling of traffic produced by adaptive HTTP-based video. In: *Multimedia Tools and Applications*, pp. 1–22 (2016)
 36. Müller, C., Timmerer, C.: A VLC media player plugin enabling dynamic adaptive streaming over HTTP. In: *Proceedings of the 19th ACM International Conference on Multimedia*, pp. 723–726 (2011)
 37. Müller, C., Lederer, S., Timmerer, C.: An evaluation of dynamic adaptive streaming over HTTP in vehicular environments. In: *Proceedings of the 4th Workshop on Mobile Video*, pp. 37–42. ACM (2012)
 38. Zink, M., Künzel, O., Schmitt, J., Steinmetz, R.: Subjective impression of variations in layer encoded videos. In: *Quality of Service-IWQoS 2003*, pp. 137–154. Springer (2003)
 39. Ni, P., Eichhorn, A., Griwodz, C., Halvorsen, P.: Fine-grained scalable streaming from coarse-grained videos. In: *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pp. 103–108. ACM (2009)
 40. Abramowitz, M., Stegun, I.A. (eds.): *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. Applied Mathematics Series. 55 (Ninth reprint with additional corrections of tenth original printing with corrections (December 1972); first ed.). Dover Publications, Washington DC, New York, United States. ISBN 978-0-486-61272-0 (1983)
 41. Lazaris, A., Koutsakis, P.: Modeling multiplexed traffic from H. 264/AVC videoconference streams. *Comput. Commun.* **33**(10), 1235–1242 (2010)
 42. Beran, J.: *Statistics for Long-memory Processes*, vol. 61. Chapman & Hall/CRC, BocaRaton (1994)
 43. Rangarajan, G., Ding, M.: An integrated approach to the assessment of long range correlation in time series data. *arXiv preprint arXiv:cond-mat/0105269* (2001)
 44. Wuertz, D., et al.: *Rmetrics*. <https://www.rmetrics.org/> (2015)
 45. R Core Team: *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna (2015)
 46. Yadav, R.K., Balakrishnan, M.: Comparative evaluation of ARIMA and ANFIS for modeling of wireless network traffic time series. *EURASIP J. Wirel. Commun. Netw.* **2014**(1), 1–8 (2014)
 47. Dethe, C.G., Wakde, D.G.: On the prediction of packet process in network traffic using FARIMA time-series model. *J. Indian Inst. Sci.* **84**(1 & 2), 31 (2013)
 48. Gadre, V., Desai, U.B., et al.: *Multifractal Based Network Traffic Modeling*. Springer Science & Business Media, Berlin (2012)
 49. Adebisi, A.A., Adewumi, A.O., Ayo, C.K.: Comparison of ARIMA and artificial neural networks models for stock price prediction. *J. Appl. Math.* **1–7**, 2014 (2014)
 50. Valipour, M., Banihabib, M.E., Behbahani, S.M.R.: Comparison of the ARMA, ARIMA, and the autoregressive artificial neural network models in forecasting the monthly inflow of Dez dam reservoir. *J. Hydrol.* **476**, 433–441 (2013)
 51. Mirzaei, M.M., Mizanian, K., Rezaeian, M.: Modeling of self-similar network traffic using artificial neural networks. In: *2014 4th International eConference on Computer and Knowledge Engineering (ICCCKE)*, pp. 741–746. IEEE (2014)
 52. Ahmed, S.: IP traffic forecasting using focused time delay feed forward neural network. *Bahria Univ. J. Inf. Commun. Technol.* **2**(1), 1 (2009)
 53. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. *Cognit. Model.* **5**(3), 1 (1988)
 54. Socher, R., Pennington, J., Huang, E.H., Ng, A.Y., Manning, C.D.: Semi-supervised recursive autoencoders for predicting sentiment distributions. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 151–161. Association for Computational Linguistics (2011)
 55. Falk, M., Marohn, F., Michel, R., Hofmann, D., Macke, M., Tewes, B., Dinges, P.: A first course on time series analysis: examples with SAS. Available online: http://statistik.mathematik.uniwuertzburg.de/wissenschaftsforschung/time_series/ (2006)
 56. Hyndman, R.J., Athanasopoulos, G.: *Forecasting: principles and practice*. OTexts (2014)
 57. Cowpertwait, P.S.P., Metcalfe, A.V.: *Introductory time series with R*. Springer Science & Business Media, Berlin (2009)
 58. Takens, F.: *Detecting Strange Attractors in Turbulence*. Springer, Berlin (1981)
 59. Rhodes, C., Morari, Manfred: The false nearest neighbors algorithm: an overview. *Comput. Chem. Eng.* **21**, S1149–S1154 (1997)
 60. Kantz, H., Schreiber, T.: *Nonlinear Time Series Analysis*, vol. 7. Cambridge University Press, Cambridge (2004)
 61. Venables, W.N., Ripley, B.D.: *Modern Applied Statistics with S*, 4th edn. Springer, New York (2002)
 62. O'Connor, P., Kleyner, A.: *Practical Reliability Engineering*. Wiley, Hoboken (2011)
 63. Lederer, S., Müller, C., Timmerer, C.: Dynamic adaptive streaming over HTTP dataset. In: *Proceedings of the 3rd Multimedia Systems Conference*, pp. 89–94 (2012)