Traffic Model for HTTP-Based Adaptive Streaming

Silvio Waldmann
Technische Universität Berlin
Berlin, Germany
silvio.mf.waldmann@campus.tu-berlin.de

Konstantin Miller
Technische Universität Berlin
Berlin, Germany
konstantin.miller@ieee.org

Adam Wolisz
Technische Universität Berlin
Berlin, Germany
awo@ieee.org

***Abstract*—The amount of video traffic on the Internet has seen a tremendous increase over the past few years. In 2020, it is predicted to account for 85% of the total Internet consumer traffic. Due to this dominant role, streaming traffic has to be considered by workload models used to evaluate the performance of networking systems. A de facto standard technology for Internet-based Video on Demand (VoD) services is HTTP-Based Adaptive Streaming (HAS), which is also increasingly used for live streaming. Unfortunately, HAS clients produce a very specific workload pattern that is not appropriately represented by traditional HTTP traffic models. In the present work, we propose a stochastic model accurately describing such traffic, along with a methodology to generate synthetic traffic using functionality provided by commonly available numerical/scientific software libraries. We perform a proof of concept by fitting the model to a data set collected in a residential Wi-Fi environment, and generating synthetic traffic matching the characteristics of the traffic in the collected data set.**

***Index Terms*—Traffic model; traffic generator; adaptive streaming; MPEG-DASH**

I. Introduction

Over the past years, we have been observing a tremendous increase in video traffic. By 2020, it is predicted to account for approximately 85% of the total consumer Internet traffic [1]. This growth is driven by several factors such as: the broad adoption of new high-speed wireless communication technologies, proliferation of powerful mobile terminals with high-resolution screens, and increasing video bit rates due to new and emerging formats such as HD, 4K, 360, or Virtual Reality (VR).

All Internet-based video services face the challenge to serve an extremely heterogeneous set of clients under extremely heterogeneous network conditions. Both the user on an otherwise idle high-speed fixed-line connection, and the fast-moving user on the edge of a busy mobile cell, should be able to receive the video representation which is most appropriate for their conditions. This has led to a wide adoption of adaptive streaming technologies, such as HAS, also known as Dynamic Adaptive Streaming over HTTP (DASH) [2], that is capable of providing a smooth viewing experience in dynamic environments. Prominent examples include MPEG-DASH [3] and Apple HLS [4].

Due to its dominating role in the overall Internet traffic mix, streaming traffic has to be considered by the workload models used when evaluating the performance of networking systems. Unfortunately, traditional HTTP traffic models do not adequately represent adaptive streaming traffic. The main reason is that with HAS, the size of the request dynamically varies as a function of the throughput, which is particularly pronounced in wireless/mobile networks that often exhibit strong throughput fluctuations. Another reason is that a streaming flow has a significantly higher data rate. Furthermore, it exhibits a pattern with a much higher degree of regularity.

Moreover, we argue that adaptive streaming traffic is best described in terms of request durations rather than request sizes. The reason is that since each request contains a fixed duration τ of video content, the client adapts the video bit rate such that on average, the download of one video segment is taking approximately $\alpha\tau$ seconds. Here, $\alpha \leq 1$ may account for several factors. First, the client may deploy a safety margin which prevents it from fully exploiting the available capacity. Second, a video bit rate matching the available capacity may not be available at the server. Finally, in the presence of throughput fluctuations, a client may need to frequently switch the video representation in order to match the average available capacity. This, however, is known to annoy the user and thus degrade the Quality of Experience (QoE). Consequently, a client may stick to the highest available video quality which is still below the available capacity, to provide a smooth viewing experience without excessive quality transitions.

In the present work, we propose a stochastic model consisting of several components describing the distributions and correlations of request completion times, and inter-segment times (time between two subsequent requests). The proposed model is a request-level model. The packet-level behavior is assumed to be governed by the Transmission Control Protocol (TCP). Consequently, the developed model can be deployed on top of a TCP model (e.g., in a simulator), or on top of an actual TCP implementation (e.g., in a traffic generator).

We specifically target Wi-Fi environments. However, the proposed methodology can be also applied to data sets collected in fixed-line and mobile environments. Furthermore, we focus on VoD traffic, which, at the moment, comprises by far the largest portion of the streaming traffic on the Internet.

Our second contribution is a method to implement the proposed model in order to generate synthetic streaming traffic. This method leverages functionality provided by commonly available numerical/scientific libraries, and thus can be easily implemented in various environments.

To perform a proof of concept, we fit the parameters of the model to a data set collected in a residential Wi-Fi environment. As streaming clients we use dash.js, which is

the reference streaming client of the DASH Industry Forum[1], and TAPAS [5]. The proof of concept confirms that the model accurately captures the essential parameters of the adaptive streaming traffic.

The rest of the paper is structured as follows. Section II describes the related work. Section III briefly reviews the principles of HAS and introduces the proposed model. Section IV proposes a methodology to synthetically generate traffic based on the developed model. A proof of concept is performed in Section V. Finally, Section VI concludes the paper.

II. Related Work

Several studies propose models for non-adaptive streaming [7]–[9]. Without adaptation, however, the size of the requested segments is independent of the throughput. In contrast, with adaptation the segment size is a function of the throughput and the deployed adaptation logic.

Summers [10] propose a model for adaptive streaming based on request sizes rather than request durations. The model is used to evaluate the performance of HTTP servers. Reed and Aikat [6] characterize mean and variance of times between segment requests, mean and variance of 10 second throughput averages, and maximum segment size received during a streaming session. Such models, however, cannot be applied to network performance evaluation since they do not reflect the dependency between the throughput and the request size. In contrast, our model characterizes request durations, inter-segment times, and their temporal correlations.

Laterman [11] characterizes request durations but not the inter-segment times such that the resulting model cannot be used to generate synthetic traffic. Also, the proposed model does not take into account temporal correlations.

Several studies focus on modelling aggregated traffic, as observed, e.g., on backbone links, or campus network edge routers [12]. In contrast, we propose a more granular per-flow model, which may be applied to model video traffic in access networks or individual streaming sessions.

Studies targeting session-level modeling include [10], [13].

Traditionally, HTTP traffic models have been developed for web traffic [14]. Web traffic, however, distinguishes itself from HTTP-based streaming traffic in several aspects. The latter exhibits a much higher degree of regularity, and the size of the request dynamically varies as a function of the throughput.

III. Modeling Approach

In this section, we briefly review the core principles of HAS, before we describe the developed traffic model in details.

With HAS, the video is split into segments, each containing a few seconds of the content. We denote the duration of a segment by τ, and assume that it remains constant throughout a streaming session. Each segment is available in multiple representations that typically differ w.r.t. the mean video bit rate. Once downloaded, the segments are stored in the playback buffer. When their playback deadline approach, they are retrieved from the buffer, decoded, and played out. During the playback, the representation—technically—may be switched at each segment boundary.

A HAS client downloads the individual video segments, typically in chronological order. For each segment, it selects the appropriate representation. One of its goals is to maximize the user's QoE, which reflects the user's satisfaction with the service. The QoE is mainly influenced by four factors. First, the number of playback interruptions that occur when a segment download is not completed prior to the playback deadline (that is, the playback buffer runs empty). Second, the average video quality. Third, the number of quality transitions. And finally, the start-up delay, which is particularly important when a user watches short videos or is browsing channels.

In order to prevent the buffer from overflowing, the client may delay the download of certain segments producing inter-request delays. For live streaming, a second reason for inter-request delays is the fact that a segment may not yet be available for download when the download of the previous segment has been completed. Thus, the generated traffic resembles an ON/OFF pattern. During the ON time, the client is downloading a segment at a rate controlled by TCP, while during the OFF time, no transmission is performed. Most clients reuse TCP connections for multiple downloads, avoiding the inefficiency of frequently creating new TCP flows. We denote the random variable representing the ON time by T_{ON}, the random variable representing the OFF time by T_{OFF}, and their sum—the inter-request time—by $T_{\text{IR}} = T_{\text{ON}} + T_{\text{OFF}}$.

The core idea underlying the modeling approach is based on the following obervations. First, an adaptive client strives to adapt the video bit rate such that the average duration of a segment download $\bar{T}_{\text{ON}}$ equals or is slightly less than the duration of the video content τ contained in a segment. Otherwise, if $\bar{T}_{\text{ON}} > \tau$, the client will encounter playback interruptions due to buffer underruns, while if $\bar{T}$ is substantially smaller than τ, the client will underutilize the available resourced resulting in a lower QoE. Thus, one essential component characterizing streaming traffic is the distribution of ON times. Note that this is in contrast to traditional HTTP traffic models that model the size of the request rather than its duration.

Second, the distribution of the inter-request times T_{IR} must be centered exactly around τ. If its mean value $\bar{T}_{\text{IR}}$ is strictly smaller than τ, the buffer level of the client will continuously increase, resulting in overflow. If it is larger, the buffer level will decrease, resulting in frequent playback interruptions.

Given the distributions of T_{ON} and T_{IR}, we may already generate an ON/OFF pattern. However, the similarity with actual streaming traffic would be poor. In addition, in order to further increase the accuracy of the model, we take into account the temporal correlation of T_{ON}. The intuition is that the ON/OFF pattern generated by the clients is both a function of the throughput and of the video bit rate within a representation, which both exhibit a high temporal correlation, at least at short lags.

Consequently, our proposed model includes the following three components: the distribution of T_{ON}, the distribution of

[1]http://dashif.org

T_{IR}, and the auto-correlation of T_{ON}.

IV. Synthetic Traffic Generation

The main envisaged application of the proposed traffic model is generation of synthetic streaming traffic for performance evaluation studies of various networking systems components. Consequently, in the following, we propose an approach on how to instantiate the model using standard functionality commonly available in various computing environments, such as Python, or Matlab.

A. Fitting the model to a data set

The first step to instantiating the model is to fit its parameter to a data set representing realistic streaming traffic in a targeted environment. Concretely, we need to estimate the distributions of T_{ON} and T_{IR}, and the auto-correlation of T_{ON}. In Section V, we will provide example values that we obtained by fitting the model to a data set collected in a residential Wi-Fi environment, using two different HAS clients.

Most computing environments offer the functionality for fitting distributions. In Python, this functionality is provided by the function `scipy.stats.rv_continuous.fit()`, while in Matlab it is provided by `fitdist()`. Both function use the Maximum Likelihood Estimation (MLE) approach [15], [16] to fit the parameters of a given distribution to the data (with a few exceptions where other methods are more appropriate).

MLE may also be applied directly. Given a candidate distribution represented its Probability Density Function (PDF) f that has parameters θ, the best fit is provided by a parameter vector θ_0 maximizing the log-likelihood

$$L_f(\theta; a, b) = \sum_{i=0}^{n-1} \log f(x_i; \theta) - n \log \left[(F(b; \theta) - F(a; \theta) \right] .$$

Here, $\{x_0, \ldots, x_{n-1}\}$ are the collected measurements (for example, the measured ON values). $[a, b]$ is the interval on which the distribution shall be fitted. If the distribution shall not be truncated, that is $[a, b]$ is the support of f, the second summand equals 0.

In order to find the value for θ maximizing L_f, one can use optimization routines such as `scipy.optimize.brute` (in case of a few parameters) or `scipy.optimize.minimize` available in Python, or `fmincon()` available in Matlab. Caution is demanded, however, since, depending on the candidate distribution, the likelihood function may not be convex and thus have local maxima. In order to compare different models, a goodness of fit test, such as the Kolmogorov-Smirnov test, may be used. Finally, it is worth noting that non-parametric approaches such as fitting a kernel distribution may be used as well.

There are many ways to compute the Autocorrelation Function (ACF). A straightforward way is to use the `statsmodels.tsa.stattools.acf()` function from the StatsModels[2] module. It can also be computed indirectly using `numpy.correlate()` or `numpy.corrcoef()` functions from the NumPy[3] module. In Matlab, this can be achieved using the `autocorr()` function.

Algorithm 1: Synthetic Traffic Generation

Input: f_{ON}, θ_{ON} ▷ *distribution and param. of request durations*
Input: f_{IR}, θ_{IR} ▷ *distribution and param. of inter-request times*
Input: R_{ON} ▷ *autocorrelation of request durations*
Input: n ▷ *number of requests to be generated*
Output: $\hat{T}_{\text{ON}}$, $\hat{T}_{\text{IR}}$ ▷ *request durations and inter-request times*

1 $\hat{T}_{\text{ON}} = \text{random}(f_{\text{ON}}, \theta_{\text{ON}}, n)$ ▷ *generate i.i.d. request durations*
2 $\hat{T}_{\text{ON}} = \frac{\hat{T}_{\text{ON}} - \mu_{\text{ON}}}{\sigma_{\text{ON}}}$ ▷ *normalize to mean 0 and STD 1*
3 $\hat{T}_{\text{ON}} = \text{filter}(\hat{T}_{\text{ON}})$ ▷ *enforce the given autocorrelation*
4 $\hat{T}_{\text{ON}} = \sigma_{\text{ON}} \hat{T}_{\text{ON}} + \mu_{\text{ON}}$ ▷ *restore mean and STD*
5 $\hat{T}_{\text{IR}} = \text{random}(f_{\text{IR}}, \theta_{\text{IR}}, n)$ ▷ *generate inter-request times*

B. Generating synthetic traffic

Now, in order to create synthetic traffic that implements the statistical properties of the model, we need to ensure that each of the three components comprising the model is reflected in the generated traffic. To draw numbers from a given distribution, one can leverage the functionality typically available in most computing environments. However, the generated sequences will be i.i.d. while we specifically require the generated T_{ON} sequence to have a particular auto-correlation, at least over small lags.

In order to generate data with the given auto-correlation, we first normalize the sequence of generated i.i.d. numbers to have a mean value of 0 and a standard deviation of 1. We proceed by transforming the resulting sequence using the Z-transform [17]. We then multiple the transformed sequence by $1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}$, where a_i is the auto-correlation coefficient at lag $i > 0$. Since the auto-correlation rapidly decays, considering only the first few values, or even just the first value, should already produce good results. This transformation can be applied by using the filter functions, e.g., `scipy.sinal.lfilter` in Python, or `filter()` in Matlab. Finally, we restore the original standard deviation and mean value of the sequence.

The traffic generation approach is described by the pseudo code in Algorithm 1. Input to the traffic generator is: the PDF f_{ON} of T_{ON} and its parameters θ_{ON}, the PDF f_{IR} of T_{IR} and its parameters θ_{IR}, the vector of auto-correlation values of T_{ON} over the first few lags R_{ON}, and the number n of requests to be generated. The output contains a vector of n values for the request durations $\hat{T}_{\text{ON}}$, and a vector of n values of inter-request times $\hat{T}_{\text{IR}}$.

V. Proof of Concept

To perform a proof of concept for the proposed model, we have used it to create synthetic traffic based on a data set collected in a residential Wi-Fi environment, using two different HAS clients.

[2] http://statsmodels.sourceforge.net/

[3] http://www.numpy.org/

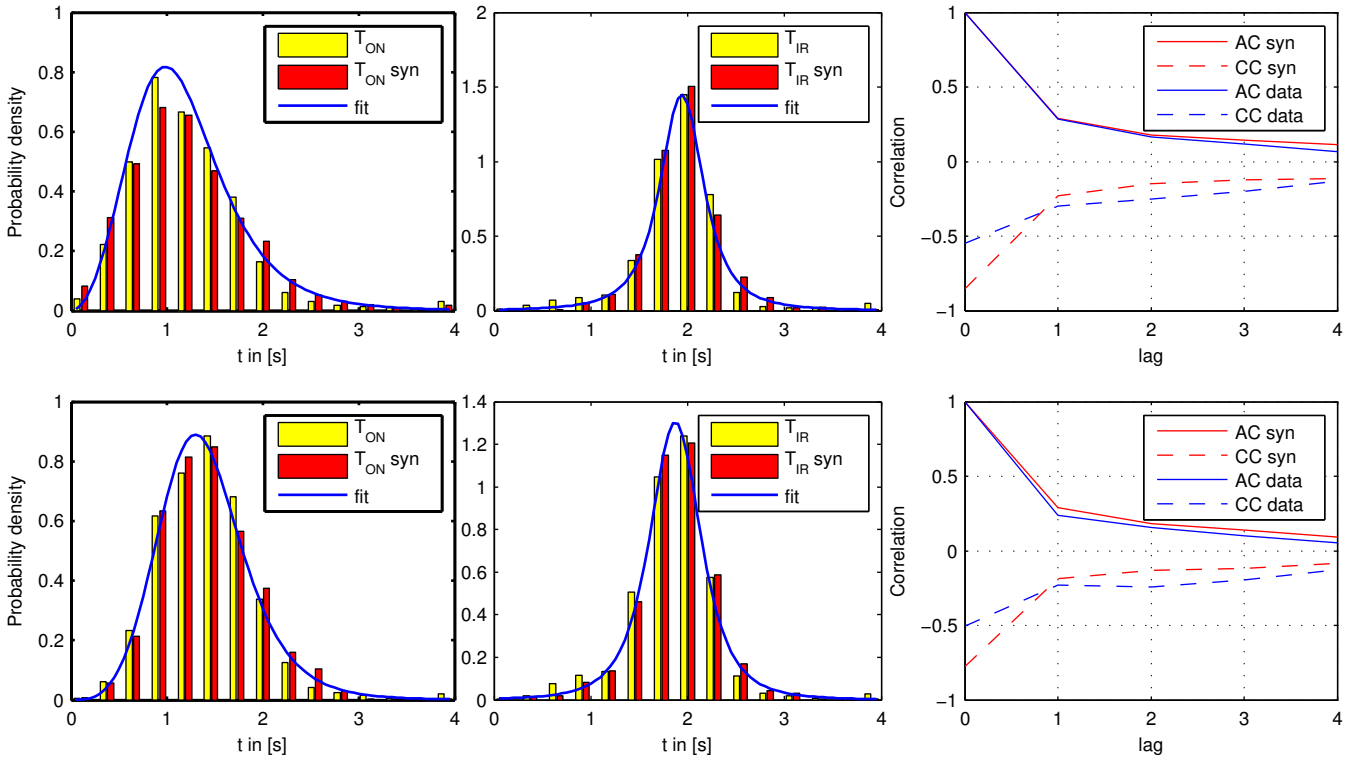

Fig. 1. Traffic parameters for dash.js without additional delay (top row) and with additional delay (bottom row). Left column shows the request durations, middle column the inter-request times, right column the autocorrelation of request durations and the cross-correlation of request durations and inter-request times.

A. Setting

As video content we have used the animated movie Sintel[4], encoded into MPEG-DASH format. We have used 2 seconds as the segment duration, and 8 representations with mean media bit rates of (in [kbps]): 91, 180, 469, 978, 2058, 3953, 9581, 21373, covering resolutions from 564 x 240 to 5076 x 2160 pixels. The bit rates were selected so as to obtain a linear increase of the Peak Signal-to-Noise Ratio (PSNR). The coefficient of variation of mean bit rates among segments of one representation is approximately 0.5 for all 8 representations.

We have used two different streaming clients: dash.js, which is the reference client of the DASH Industry Forum, and TAPAS [5] with the default adaptation algorithm as described in [18].

The measurements were performed in a residential wireless network in Berlin, Germany. As server, we used a host located on the campus of the Technische Universität Berlin. In order to capture a potential impact of a high Round-Trip Time (RTT), each measurement was repeated with an additionally introduced normaly distributed one-way delay in the server's downlink with a mean of 80 ms and a standard deviation of 20 ms. To introduce the delay, we used tc-netem [19].

For each client, 25 runs have been performed with additional delay, and 25 runs without. Each run lasted for 10 minutes. In total, a data set containing 1000 minutes, or approximately 16 hours, of data traces has been collected.

[4]https://durian.blender.org/

B. Results

To assess the accuracy of the proposed model, we compare the statistical parameters of the collected data set with those of the synthetically created traffic. Concretely, we compare the distribution of the ON times, of the inter-request times, the auto-correlation of the ON times, and the cross-correlation between the ON times and the inter-request times.

The results are illustrated in Figures 1 (for dash.js) and 2 (for TAPAS). In both figures, the top row represents measurements without additional delay, while the bottom row represents measurements with additional delay. In addition, Table I shows the parameters of the fitted distribution for the ON times, while Table II shows the parameters for the inter-request times.

The first column shows the histograms of T_{ON} from both measured and synthesized data. In addition, it shows a distribution which was selected as the best fit to the data. In this case, it was the Burr distribution with parameters shown in Table I. The second column shows the histograms of the inter-request times T_{IR}. In this case, the best fit was achieved by the t distribution. The corresponding parameters are shown in Table II. In addition, both tables show the parameters obtained by fitting the same type of distribution to the synthetically generated trace.

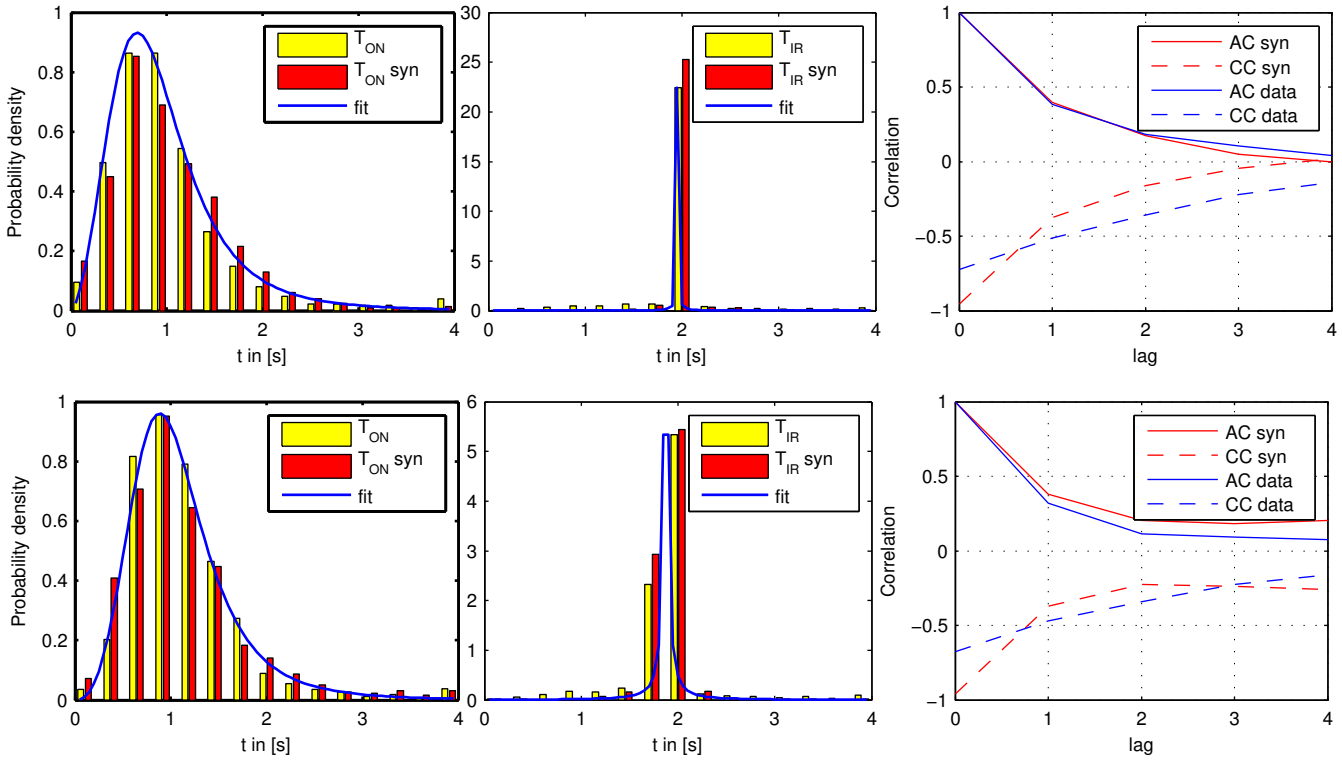

Fig. 2. Traffic parameters for TAPAS without additional delay (top row) and with additional delay (bottom row). Left column shows the request durations, middle column the inter-request times, right column the autocorrelation of request durations and the cross-correlation of request durations and inter-request times.

TABLE I
FITTED PARAMETERS OF THE BURR DISTRIBUTION FOR T_{ON}

	α	k	c
dash.js without delay	1.469	1.915	3.014
Synthetic	1.535	2.109	2.782
dash.js with delay	1.647	1.828	4.158
Synthetic	1.781	2.201	3.819
TAPAS without delay	1.033	1.451	2.671
Synthetic	1.477	2.317	2.001
TAPAS with delay	1.096	1.214	3.488
Synthetic	1.471	2.062	2.617

TABLE II
FITTED PARAMETERS OF THE T DISTRIBUTION FOR T_{IR}

	μ	σ	ν
dash.js without delay	1.938	0.245	2.086
Synthetic	1.951	0.252	2.794
dash.js with delay	1.872	0.280	2.790
Synthetic	1.858	0.305	3.778
TAPAS without delay	1.953	0.001	0.402
Synthetic	1.953	0.0011	0.456
TAPAS with delay	1.875	0.020	0.670
Synthetic	1.876	0.022	0.834

It can be seen that the inter-request times are clustered around the segment duration $\tau = 2$, as expected. The ON times cluster in the interval $[0, \tau]$. From the distribution of ON times, it can be concluded that TAPAS implements a more conservative approach than dash.js, since the center of the cluster is further to the left.

It can also be seen that the introduction of an additional delay slightly shifts the distribution of ON times to the right, due to the decreased application layer throughput.

Finally, from the broader distribution of inter-request times produced by the dash.js player, we may conclude that it is more inclined to download segments in bursts, while having longer delays between the bursts. In contrast, the TAPAS player seems to more regularly start a new segment download—every τ seconds most of the times.

The last columns in Figures 1 and 2 show the auto-correlation of ON times, and the cross-correlation between ON times and inter-request times over the lags 0 to 4. It can be seen that the filtering approach accurately reproduces the auto-correlation of ON times seen in the original data. In addition, the cross-correlation between the ON times and the inter-request times, even though not explicitely taken into account by the proposed model, is reproduced by the model with a high degree of accuracy.

Figure 3 visualizes an example run (left column), and a synthetic run (right column).

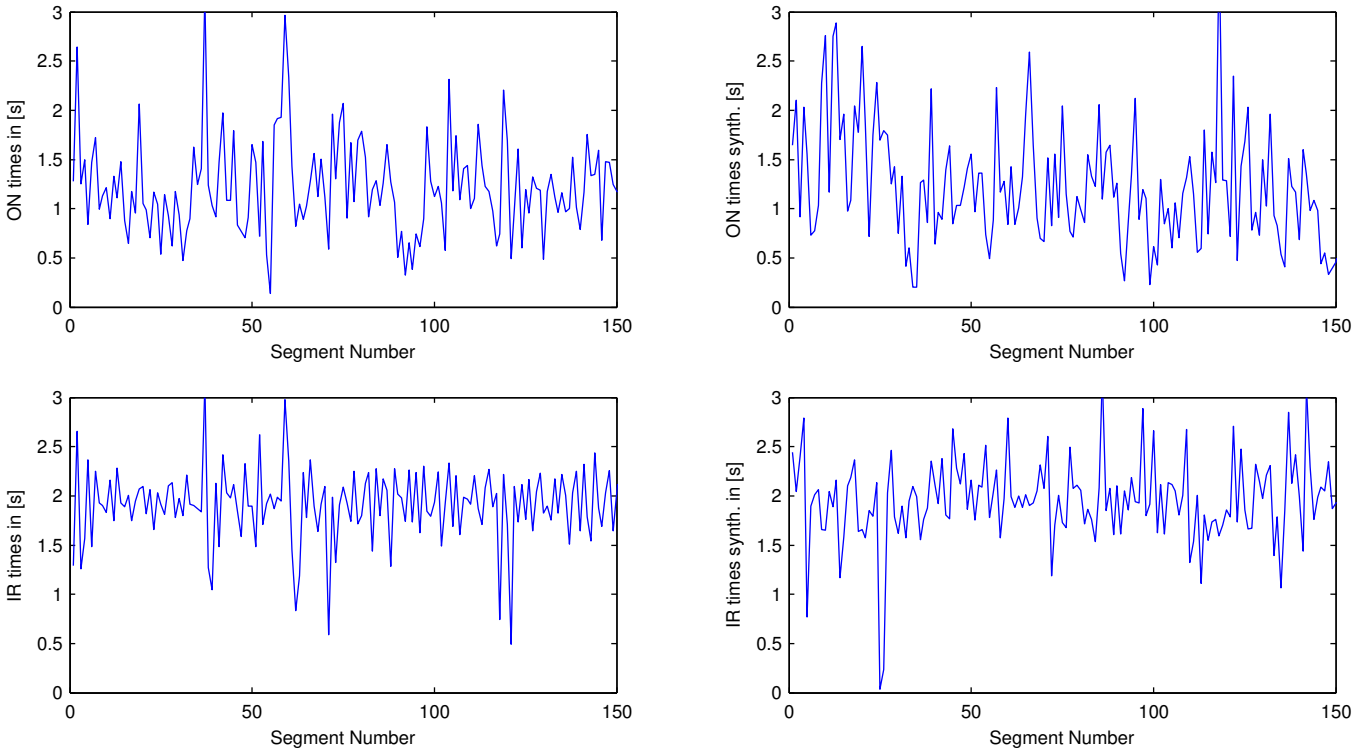

Fig. 3. Example run. Original data (left column) and synthetic data (right column).

VI. Conclusion

In the present work, we proposed a stochastic model for adaptive HTTP-based video streaming traffic. In contrast to traditional HTTP traffic models, it describes the requests in terms of durations rather than in terms of request sizes. We have fitted the parameters of the model based on a data set collected in a residential Wi-Fi network, using two adaptive streaming clients. The proposed model can be used to generate workload for experiments targeting a performance evaluation of various networking mechanisms and protocols.

Future work focuses on more advanced modeling techniques that may improve the accuracy of the model, as well as explicitely capture the dependency between throughput dynamics and the dynamics of the generated synthetic traffic.

References

[1] "Cisco Visual Networking Index: Forecast and Methodology, 2015 - 2020," Cisco Systems, Inc., White Paper, 2016.

[2] I. Sodagar, "The MPEG-DASH Standard for Multimedia Streaming Over the Internet," *IEEE Multimedia*, vol. 18, no. 4, pp. 62–67, 2011.

[3] "Dynamic Adaptive Streaming over HTTP (DASH), First Edition," International Standard ISO/IEC 23009, 2012.

[4] R. Pantos and W. May, "HTTP Live Streaming, version 20," *IETF Informational Internet-Draft*, 2016.

[5] L. D. Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo, "TAPAS: a Tool for rApid Prototyping of Adaptive Streaming algorithms," in *Proc. of the 2014 Workshop on Design, Quality and Deployment of Adaptive Video Streaming (VideoNext)*, 2014.

[6] A. Reed and J. Aikat, "Modeling, Identifying, and Simulating Dynamic Adaptive Streaming over HTTP," in *Proc. of the IEEE International Conference on Network Protocols (ICNP)*, 2013.

[7] P. Gill, M. F. Arlitt, Z. Li, and A. Mahanti, "YouTube Traffic Characterization: a View from the Edge," in *Proc. of the ACM SIGCOMM Internet Measurement Conference (IMC)*, 2007.

[8] A. Rao, Y.-S. Lim, C. Barakat, A. Legout, D. Towsley, and W. Dabbous, "Network Characteristics of Video Streaming Traffic," in *In Proc. of ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, 2011.

[9] J. J. Ramos-Munoz, J. Prados-Garzon, P. Ameigeiras, J. Navarro-Ortiz, and J. M. Lopez-Soler, "Characteristics of Mobile YouTube Traffic," *IEEE Wireless Communications*, vol. 21, no. 1, pp. 18–25, 2014.

[10] J. A. Summers, "Understanding and Efficiently Servicing HTTP Streaming Video Workloads," Ph.D. Thesis, University of Waterloo, 2016.

[11] M. Laterman, "NetFlix and Twitch Traffic Characterization," Master Thesis, University of Calgary, 2015.

[12] A. Biernacki, "Analysis of Aggregated HTTP-Based Video Traffic," *Journal of Communications and Networks*, vol. 18, no. 5, pp. 826–836, 2016.

[13] A. Finamore, M. Mellia, M. M. Munafò, R. Torres, and S. G. Rao, "YouTube Everywhere: Impact of Device and Infrastructure Synergies on User Experience," in *Proc. of the ACM SIGCOMM Internet Measurement Conference (IMC)*, 2011.

[14] R. Pries, Z. Magyari, and P. Tran-Gia, "An HTTP Web Traffic Model Based on the Top One Million Visited Web Pages," in *Proc. of Conference on Next Generation Internet (NGI)*, 2012.

[15] I. J. Myung, "Tutorial on Maximum Likelihood Estimation," *Journal of Mathematical Psychology*, vol. 47, no. 1, pp. 90–100, 2003.

[16] J.-Y. L. Boudec, *Performance Evaluation of Computer and Communication Systems*. EFPL Press, 2015.

[17] A. V. Oppenheim and R. W. Schaefer, *Discrete-Time Signal Processing*, 3rd ed. Pearson, 2009.

[18] Z. Li, X. Zhu, R. Pan, H. Hu, A. Begen, and D. Oran, "Probe and Adapt: Rate Adaptation for HTTP Video Streaming at Scale," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 719–733, 2014.

[19] L. Nussbaum and O. Richard, "A Comparative Study of Network Link Emulators," in *Proc. of the Spring Simulation Multiconference (SpringSim)*, 2009.