JID: COMCOM [m5G;April 11, 2016;10:19]

Computer Communications xxx (2016) xxx–xxx

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Modeling live adaptive streaming over HTTP

Q1 Savera Tanwir*, Harry Perros

Q2 *Department of Computer Science, North Carolina State University, Raleigh, NC, United States*

A R T I C L E I N F O

Article history:
Received 7 December 2015
Revised 24 February 2016
Accepted 30 March 2016
Available online xxx

Keywords:
HTTP streaming
DASH
Live streaming
Analytical model
Rate adaptation algorithm

A B S T R A C T

Video streaming methods have evolved greatly over the years. Today, the most prevalent technique to stream live and video on-demand is the adaptive HTTP streaming and is used by several commercial vendors. In this paper, we present an approximate analytic model for live adaptive streaming over HTTP. Using this model, we propose a new rate control algorithm that makes the rate transitions less frequent and increases the quality of experience for the viewer. Also, the model can be used to characterize the departure packet process at the video server. To the best of our knowledge, this is the first video traffic model for adaptive HTTP streaming to be reported in the literature.

1. Introduction

Over the last few years video-based applications, and video streaming in particular, have become very popular generating more than half of the aggregate Internet traffic [1]. This has become possible through the gradual development of highly efficient video compression methods, broadband access technologies, QoS schemes in the IP network and the development of adaptive video players. Today, the most popular and cost effective means for video streaming is adaptive streaming over HTTP. Multimedia content can now be delivered efficiently in larger segments using HTTP. The basic idea is to chop a continuous stream into segments, encode these in multiple qualities and make these available for download using plain HTTP methods. The client video player application monitors the download speed and requests chunks of varying quality in response to changing network conditions. The main advantage of HTTP based streaming is that the deployed web infrastructure is easily reused, even for live segment streaming. In case of live streaming, the segments are produced periodically; with a new segment becoming available shortly after it has been recorded and encoded completely.

Several recent players, such as Microsoft Smooth Streaming, Apple's HTTP Live Streaming, Adobe OSMF and Netflix players all use adaptive streaming over HTTP. However, each implementation uses formats and proprietary client protocols. Due to the market prospects and requests from the industry, adaptive streaming has been standardized by 3GPP and ISO as MPEG-DASH (Dynamic Adaptive Streaming over HTTP) in 2011 [2]. In addition to providing all benefits of streaming over HTTP, DASH supports live media services and it is bitrate adaptive.

Different aspects of dynamic adaptive HTTP streaming have been explored in the literature. The research work done in the area of adaptive HTTP streaming is mostly focused on the performance and design of efficient rate control algorithms and the interactions of HTTP streaming with TCP. However, there is a lack of analytical models for video streaming traffic over HTTP. Performance modeling is necessary for service providers to properly maintain quality of service (QoS) and it requires accurate traffic models that have the ability to capture the statistical characteristics of the actual traffic on the network. Better understanding of the network through modeling provides the means to make better design decisions. In this paper, we present the first (to the best of our knowledge) analytic model for live adaptive streaming over HTTP. Using this model, we propose a new rate control algorithm that reduces the number of rate transitions and increases the quality of experience for the viewer. The proposed model can also be used to characterize the departure packet process at the video server.

This paper is organized as follows. In Section 2, we summarize the research done in this area. In Section 3, we present our model and in Section 4 we provide a validation of its accuracy. In Section 5 we describe a new rate control algorithm based on the proposed analytic model. Lastly, the summary is presented in Section 6.

2. Literature review

Different aspects of dynamic adaptive HTTP streaming have been explored in the literature over the past few years. Several

* Corresponding author. Tel.: +19199098545.
E-mail address: saveratanwir@gmail.com, stanwir@ncsu.edu (S. Tanwir).

http://dx.doi.org/10.1016/j.comcom.2016.03.025
0140-3664/

Please cite this article as: S. Tanwir, H. Perros, Modeling live adaptive streaming over HTTP, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.025

performance studies have been conducted to compare various players that use adaptive HTTP streaming. In [3], Akhshabi et al. conducted an experimental evaluation of three commercial adaptive HTTP streaming players, i.e., Microsoft Smooth streaming, Netflix and Adobe OSMF player. They noted that all players had their shortcomings and further research is needed in order to improve the rate adaptation algorithms. A study of the performance of Adaptive HTTP Streaming over different access networks is presented in [4]. Muller et al. compared Microsoft Smooth Steaming (MSS), Adobe HTTP Dynamic Streaming (HTS), and Apple HTTP Live Streaming (HLS) and DASH in a vehicular environment in [5], using the client implementations for the proprietary systems and their own DASH client. In [6], Miller et al. compare MSS client and their own DASH client in Wireless Local Area Network (WLAN) environment. In [7], the different delay components in DASH for live streaming are identified and analyzed. The best performance in terms of reduced delay is obtained with short media segments but short segments increase server load. Seufert et al. surveyed the literature that covers QoE aspects of adaptation dimensions and strategies in [8]. They reviewed recent developments in the field of HTTP adaptive streaming (HAS), and existing open standardized and proprietary solutions.

Several rate adaptation algorithms and optimization strategies have been proposed in the literature for adaptive video streaming over HTTP. In [9], Miller et al. presented an algorithm that aims at avoiding interruptions of playback, maximizing video quality, minimizing the number of video quality shifts and minimizing the delay between user's request and the start of the playback. Tian and Liu proposed a rate control algorithm [10] that smoothly increases video rate as the available network bandwidth increases, and promptly reduces video rate in response to sudden congestion events. In [11], Bokani et al. consider a Markov Decision Process (MDP) to derive the optimum segment rate selection strategy that maximizes streaming quality. Xing et al. [12] also formulated the optimal video streaming process with multiple links as a Markov Decision Process (MDP). MDP is time consuming and computationally expensive, and in view of this they also proposed an adaptive, best-action search algorithm to obtain a sub-optimal solution. Mansy et al. [13] proposed a technique called SABRE (Smooth Adaptive Bit RatE), that enables a video client to smoothly download video segments from the server without causing significant delays to other traffic sharing the link. In [14], Liu et al. proposed two new rate adaptation algorithms for the serial and the parallel segment fetching methods. Jiang et al. proposed a rate adaptation algorithm called FESTIVE (Fair, Efficient, Stable, adaptIVE) in [15]. SVC has been shown as better encoding method for adaptive streaming and several authors have proposed rate adaptive algorithms for SVC encoded video in [16–18] and [19].

Apart from the research on performance and rate adaptation, the interactions of HTTP adaptive streaming with TCP has also been studied in the literature. Different aspects like fairness, TCP throughput and traffic shaping have been considered. In [20], Akhshabi et al. described how the competition for available bandwidth between multiple adaptive streaming players can lead to instability, unfairness, and bandwidth underutilization. The authors identified that once the playback buffer size reaches a certain target buffer, the player switches to the steady-state during which it aims to maintain a constant playback buffer size. The player requests one chunk every T seconds (if the download duration is less than T) or as soon as the previous chunk is received. This leads to an activity pattern in which the player is either ON, downloading a chunk, or it is OFF, staying idle. They conducted experiments with real adaptive streaming players and showed that the three issues mentioned above i.e., instability, unfairness, and bandwidth underutilization, can arise in practice. They also showed that different factors like the duration of ON-OFF periods, the fair share relative to the available profile bitrates, and the number of competing players, can affect the stability of the system. Esteban et al. examined the interactions between HTTP Adaptive Streaming (HAS) and TCP in [21]. A TCP transfer can be divided into 3 phases, the initial burst, ACK clocking, and trailing ACK phases. HAS requests are relatively small and a significant portion of the transmission duration is spent in the initial burst and trailing ACK phases. The authors note that if the congestion window is large enough and the data is small enough, the entire transmission occurs during the initial burst, eliminating the ACK clocking phase.

There is also some research done on modeling different aspects of adaptive streaming. Wang et al. [22] investigated the relationship between the capacity and responsiveness of HTTP adaptive streaming under different segment sizes and media encoding rates. Through experiments, they find that the maximum capacity can be achieved by choosing different segmentation time intervals specific to each media encoding rate. Kleinrouweler et al. [23] proposed an analytical performance model that estimates the rate at which HAS players switch quality. They have modeled the starting and stopping players as a Markov process instead of the download of individual segments. The results show that the model underestimated the average bitrate when compared with experimental runs using a proxy server. The proxy server can be placed in the gateway, or another similar network device between player and server. At the proxy server, HTTP traffic was monitored to detect starting and stopping players. In [24], Mitra and Swaminathan proposed a buffer model for the client that uses tunable buffer parameters such as sizes, thresholds, and rate of flow of data. They analyzed of the effects of thresholds and rate of movement of data among and proposed a strategy to design the buffers based on these constraints. The model is not applicable to live adaptive streaming. Chen et al. [25] propose model to predict the time-varying subjective quality (TVSQ) of rate-adaptive videos that are transported over HTTP. The TVSQ is a time series or temporal record of one or more viewers' judgments of the quality of the video as it is being played and viewed. The accuracy of the model was validated on a database of four video sequences. The estimated TVSQs can then be used to guide online rate-adaptation strategies towards maximizing the QoE of viewers. The results showed that the predicted TVSQ correlated with the measured TVSQ in subjective studies.

3. The proposed model

In this paper, we propose a novel analytical model for live adaptive streaming over HTTP. To the best of our knowledge, this is the first such analytic model for adaptive video streaming.

The model consists of the following three components:

1. The video server model
2. A queueing network model of the IP network between the client and server
3. The client video player model

In DASH, HTTP servers and HTTP caches are used to host and distribute continuous media content and the clients can access media resources through an HTTP-URL. In live adaptive streaming, the sequence of media segments is created on the fly from a continuous media stream. The segmenter function of the video server creates a new media segment every t seconds. Thus, each media segment contains t seconds worth of media data, i.e., the playback time for each segment is t seconds. The DASH Media Presentation Description (MPD) describes all available and not-yet available media segments either for the entire live session or up to the next MPD update. The client obtains the start time of the live stream from the MPD and synchronizes itself with the server. The client must be time synchronized with the server. If it is properly synchronized, it can calculate the latest available media segment

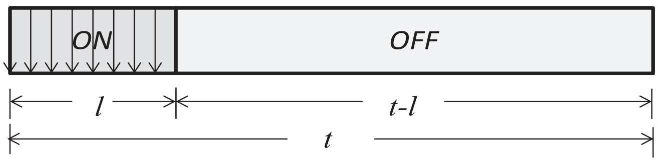

Fig. 1. Segment on-off periods.

on the server given the segment duration. It then starts fetching the media segments as they become available on the server every t seconds. The client also monitors the network bandwidth fluctuations continuously and chooses the subsequent segments accordingly.

We note that the video server transmits a segment in a series of IP packets set to Maximum Transfer Unit (MTU). The length of the segment in bytes is determined by the bitrate requested by the client. Therefore, each bitrate will have a corresponding segment size. Since all packets are equal to MTU, except the last one, we assume that the last one is also equal to MTU. This assumption does not affect the accuracy of the model, since the last packets account for a small percentage of all the transmitted packets, and permits us to define all three models in discrete-time, where the length of the time slot is equal to the amount of time it takes to transmit one IP packet of size equal to the MTU.

3.1. The video server

The nature of network traffic generated by live segment streaming is very different from the traditional bulk transfer traffic stemming from progressive video download and file transfer. The video traffic generated by the video server is determined by the client request strategy. The client downloads the segments of a stream one after another. It chooses the bitrate of the segments according to the available bandwidth so that the time it takes to download a segment is shorter than or equal to the actual segment duration (the playout time of a segment). The download time must be shorter or equal to the segment duration, otherwise the client buffer would eventually become empty and pauses would occur in the playout. In general, it takes less time to download a segment than it takes to playout the segment, i.e., the download speed is higher than the playout speed. The client buffer hides this inequality by buffering every segment that is downloaded. These successive download-and-wait operations create an on-off traffic pattern of IP packets.

Based on this observation, we have modeled the video server as an on-off video traffic source. The server transmits the packets in a video segment back to back during the on period and then stops transmitting during the off period. All packets are of equal size set to the MTU. The transmission begins again when it receives the next HTTP GET request from the client for the next video segment. In case of live-streaming, the sum of the on and off periods is always the segment duration, t, as shown in Fig. 1.

The length of the on period, l, and consequently of the off, $t-l$, period can vary throughout the life time of the connection depending on the bitrate requested by the client. The requested bitrate differs due to the variations in the available bandwidth as measured by the client. The length of the on period depends on the size of the video segment which is determined by the requested bitrate. Hence, for each video streaming rate, there will be a different length of the on-off period.

We assume that the TCP congestion window is large enough so that all the packets in a segment can be sent back-to-back in a burst. We have not modeled any TCP retransmissions that may occur due to congestion and packet loss. Retransmitted packets are of no use to the client in the case of live streaming since it maintains a buffer of one video segment only. Any packets received from previous segments are discarded. Also, we assume that the congestion control algorithm of TCP is tailored to live streaming, which means that the congestion window size is not decreased drastically during congestion, because it can cause large packet delays that can make the entire segment reach the client over a span of more than one segment durations, thus causing the client to freeze.

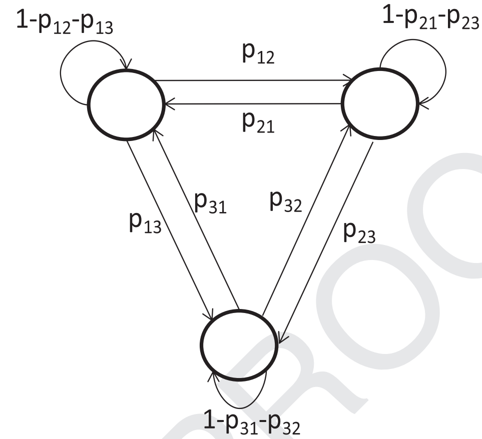

Fig. 2. Markov chain for three bitrates.

In view of these observations, we model the video source model as a Markov chain with unit time equal to video segment duration t. The states of the Markov chain represent the different qualities or bitrates that are available for download for each video. A model for three different bitrates is shown in Fig. 2. Within each state, the packets are generated using an on-off process. The length of the on period, l, is equal to the (size of the segment in a given quality)/(transmission speed of the server). The off period is t minus the length of on period. Thus, the lengths of on and off periods are fixed for each state.

In the real system, the transitions among the state of the Markov chain are caused by the client and they depend on the available bandwidth as measured by the client along with the client buffer occupancy level. Specifically, the client estimates the available bandwidth as the (segment size in bytes)/(download time for the entire segment) and subsequently it decides whether to switch to a higher or lower rate or stay at the same rate. Consequently, the transition probabilities are obtained by modeling the behavior of the client. In order to determine the client's decision as to whether to change the bitrate, we need to model the delay that the packets of the same segment suffer until they reach the client, and also how spread out these packets are from each other due to interleaving with other packets in the routers along the path from the video server to the client. This is done using the queueing network model described below.

3.2. The queueing network

We use a discrete-time queueing network to depict the network between the video server and the client. We assume that this is a wide area network (WAN) connected to an access network which serves the client. We assume that Differentiated Services (Diffserv) is used to support QoS in the network.

Differentiated Services is a multiple service scheme that provides different QoS to different flows. Several QoS classes have been defined, known as the DiffServ Code Points (DSCP). The DSCP is carried in the IP header of each packet and it is used to determine which priority queue the packet will join at the output port of a router. Video packets are typically given an AF41 priority. Consequently, the WAN is modeled by a series of single-server queues

Please cite this article as: S. Tanwir, H. Perros, Modeling live adaptive streaming over HTTP, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.025

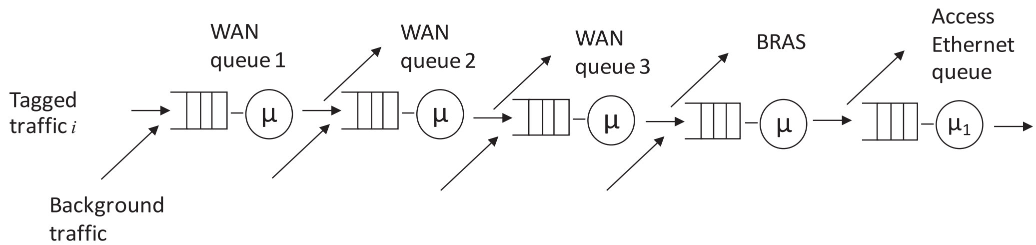

Fig. 3. The queueing network under study.

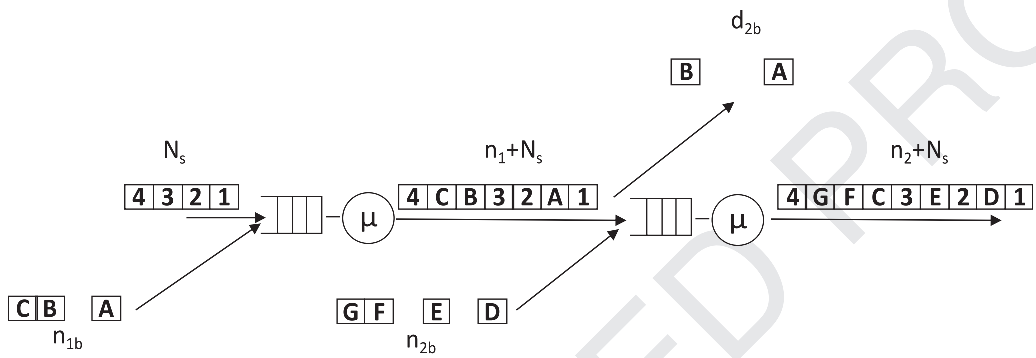

Fig. 4. Formation of the spread.

which represent the AF41 queue at the output port of each router along the path of the video stream. An example of this queueing network is shown in Fig. 3, where the first four queues represent the WAN and the last queue represents the access network. Each WAN queue receives packets transmitted from the video server to the client (tagged traffic), along with other video packet traffic from other sources (background traffic).

All packets are assumed to be equal to 1500 bytes (the path MTU). All packets in each WAN queue are served in a FIFO manner at a rate μ equal to (1500 bytes)/(speed of the link), where the link speed is the same for the four WAN queues. The background traffic in a WAN queue is transmitted to the same next hop router as the tagged traffic and it may get dispersed to different output ports of the router. It is likely though, that some part of it will be transmitted out of the same output port of the next hop router as the tagged traffic. In view of this, we assume that for each WAN queue 80% of all the background traffic that arrives at the queue departs from the queueing network after it is served and the remaining 20% continues on to the next queue (these percentages can be readily varied in the model). A similar assumption holds for the remaining WAN queues.

The last queue of the queueing network depicts part of a metro Ethernet access network. In this case, the traffic gets fanned out to the Ethernet switches, and eventually to the users. We are only modeling the first hop between the Broadband Remote Access Server (BRAS) router and an Ethernet switch. The BRAS sits at the core of an ISP's network, and aggregates user sessions from the access network. (Other hops within the access network can be easily modeled). There is no background traffic at the Ethernet switch and the service rate is $\mu_1 =$ (1500 bytes)/(speed of the link). We assume that the link speed of the Ethernet switch is a hundred times less than the WAN router link speed (other speeds can also be modeled). Due to the fan out of the traffic to the end users, we assume that 95% of the background traffic that enters from the BRAS queue follows a different path after it leaves the Ethernet switch. That is, a small percentage goes along with the tagged traffic to the user.

Of interest to the overall model proposed in this paper, are the following two quantities:

1. The spread of the original video segment transmitted by the video server, when it arrives at the client
2. The end-to-end delay in the network of the leading packet of a segment.

As will be seen, these two quantities are used in the client model presented in Section 3.5.

3.3. Calculation of the spread

Let N_s be the number of packets that make up one video segment at a given bit rate. We assume that these packets arrive back-to-back at queue 1, one per time slot, where a time slot is equal to the time it takes to transmit a 1500-byte packet. At the same time it is possible that there may be background arrivals. Background traffic enters the router from other input ports and they end up being interleaved in between the packets of the segment at the AF41 queue at the output port of the router. These packets increase the length of the original segment, i.e., they increase the amount of time elapsed between the arrival of first packet and the last packet of the video segment, referred to as the "spread".

Fig. 4 shows how the spread is formed. Let us assume that the segment consists of four packets (1,2,3,4) and during its arrival to queue 1, three background packets arrive (A,B,C). A possible formation of the spread is 4CB32A1. At the next queue, packets A and B depart and their slots are taken over by new background packets D and E resulting in a new formation 4GFC3E2D1.

As shown in Fig. 5, let n_{ib} be the number of packets that arrive during the time it takes for the spread to arrive at queue i, and let d_{ib} be the number of background packets in the spread that depart before the segment joins queue i. The remaining background packets in the spread is indicated by n_i, i.e., $n_i = n_{i-1} + n_{ib} - d_{ib}$. In the case of the access Ethernet queue $n_{ib} = 0$.

We assume a binomial distribution of the background arrival process. That is, there is a probability p that a background packet arrives in a time slot. Consequently, the probability that k background packets arrive in the first queue during the time the N_s

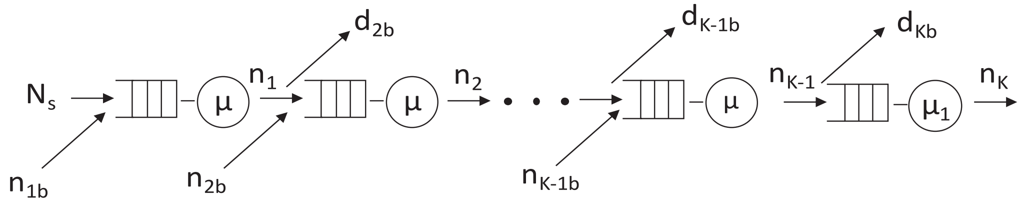

Fig. 5. The queueing network under study.

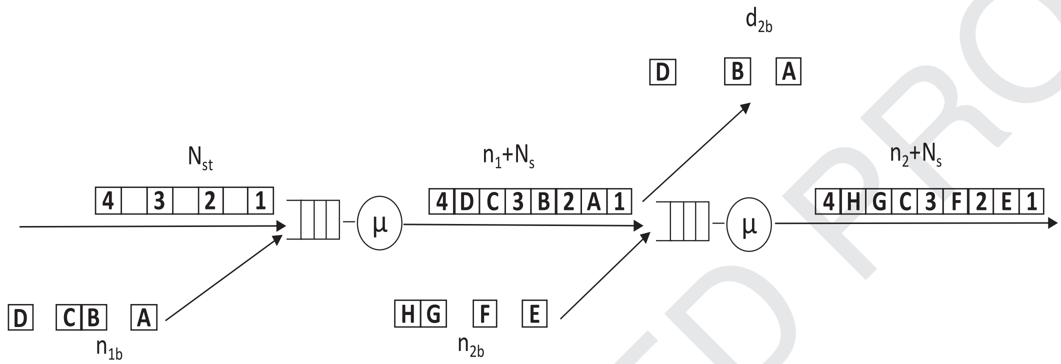

Fig. 6. Formation of the spread.

packets arrive is:

$$P[n_{1b} = k] = \binom{N_s}{k} p^k (1-p)^{N_s - k} \tag{1}$$

The probability distribution of the background packets n_2 is a convolution of n_1, the background traffic at node 2, n_{2b} and the departures at node 2, d_{2b}. Let q be the probability that a background packet leaves before the segment joins queue i. This can be written as:

$P[n_2 = l | n_1] = P[n_{2b}] \otimes P[n_1 - d_{2b}]$

or,

$P[n_2 = l | n_1] = \sum_{(j=0)}^{l} P[n_{2b} = j] P[n_1 - d_{2b} = l - j]$, if $n_1 \geq l$

and,

$P[n_2 = l | n_1] = \sum_{(l-n_1)}^{l} P[n_{2b} = j] P[n_1 - d_{2b} = l - j]$, if $n_1 < l$

where $n_1 = n_{1b}$,

$P[n_{2b} = j] = \binom{n_1 + N_s}{j} p^j (1-p)^{n_1 + N_s - j}$ and,

$P[n_1 - d_{2b} = l - j = m] = \binom{n_1}{m} q^m (1-q)^{n_1 - m}$

Unconditioning on n_1, we obtain an expression for the distribution of n_2:

$P[n_2 = l] = \sum_{n_1} (P[n_{2b}] \otimes P[n_1 - d_{2b}]) P(n_1)$

In general for queue i, we have:

$$P[n_i = l] = \sum_{n_{i-1}} (P[n_{ib}] \otimes P[n_{i-1} - d_{ib}]) P(n_{i-1}) \tag{2}$$

where,

$P[n_{ib} = j] = \binom{n_{i-1} + N_s}{j} p^j (1-p)^{n_{i-1} + N_s - j}$ and,

$P[n_{i-1} - d_{ib} = l - j = m] = \binom{n_{i-1}}{m} q^m (1-q)^{n_{i-1} - m}$

At the last queue, we do not consider any new background traffic as explained above. The distribution of the background packets can be expressed as:

$$P[n_K = l] = \sum_{n_{K-1}} (P[n_{K-1} - d_{Kb} = l]) P(n_{K-1}) \tag{3}$$

The total length of the spread is equal to the sum of the video segment packets and the background traffic packets at the last queue. Since, the video segments packets are fixed for a given bitrate, the pdf of the spread is the same as the pdf of the background traffic given by Eq. (3).

The case of slow video server

In this section we consider the case where the video server transmits packets at a rate lower than its transmission speed. This situation can arise, for instance, if it is multiplexing the video packets for multiple clients or if there are restrictions on server transmission rate from the TCP or application layer. In this case the packets that make up a segment will not be transmitted back to back. They will be spaced out and the segment will span a larger number of time slots than in the above case. We have modeled this as follows:

Let N_{st} be the number of slots that make up one video segment for a given bit rate. We assume that the video packets arrive at queue 1, one per M time slots. Let N_s denote the number of packets per segment. At the same time there may be background arrivals. Background traffic enters the router from other input ports and they are interleaved in between the packets of the segment at the AF41 queue of the output port. The background packets may fill the empty slots in between the slots occupied by the packets from the video segment. Depending on the rate of background traffic, if the background packets that arrive during N_{st} slots is more than the empty slots they will increase the spread otherwise the length of the spread remains the same at the output port of the router.

Fig. 6 shows how the spread is formed. Here we assume that the video server sends out packets at half of the link transmission speed. Let us assume that the segment consists of four packets (1,2,3,4) and during its arrival, four background packets arrive (A,B,C,D). A possible formation of the spread is 4DC3B2A1. At the next queue, packets A, B and D depart and their slots are taken over by new background packets E, F and G resulting in a new formation 4HGC3F2E1.

In this case, the number of background arrivals at queue i can be expressed as:

$P[n_i = l] = \sum_{n_{i-1}} (P[n_{ib}] \otimes P[n_{i-1} - d_{ib}]) P(n_{i-1})$

where,

$P[n_{ib} = j] = \binom{n_{i-1} + N_{sp}}{j} p^j (1-p)^{n_{i-1} + N_{sp} - j}$, and

$P[n_{i-1} - d_{ib} = l - j = m] = \binom{n_{i-1}}{m} q^m (1-q)^{n_{i-1} - m}$

Please cite this article as: S. Tanwir, H. Perros, Modeling live adaptive streaming over HTTP, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.025

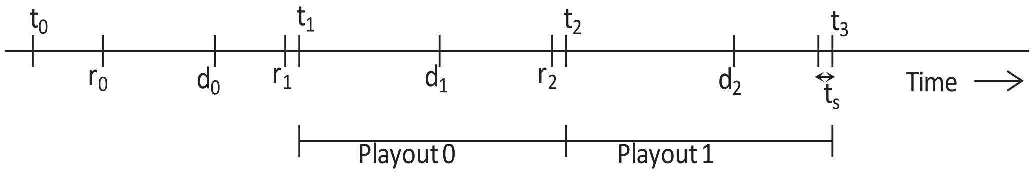

Fig. 7. Client request strategy.

where N_{sp} is the length of the spread at the input queue in terms of number of slots and is given as:

$N_{sp} = Max(N_{st}, n_{i-1} + N_s)$

In this case, the pdf of the spread is same as the pdf of background packets only if the sum of video packets and background traffic is greater than the total number of slots in the spread, N_{st}. Otherwise, the length of spread is fixed and equals N_{st}.

At the last queue, we do not consider any new background traffic. Also the spread shrinks and any empty slots disappear because of the much lower transmission speed of the last router. The distribution of background packets can be expressed as:

$P[n_K = l] = \sum_{n_{K-1}} (P[n_{K-1} - d_{Kb} = l])P(n_{K-1})$

This also gives the pdf of the number of packets in the spread.

$$P[n_K = l + N_s] = \sum_{n_{K-1}} (P[n_{K-1} - d_{Kb} = l])P(n_{K-1}) \qquad (4)$$

3.4. Calculation of the end-to-end delay

In order to calculate the total time t_e taken to download a complete video segment, we need to know the end-to-end delay of the first packet in the video segment along with the time delay between the first packet and the last packet t_{sp}. The pdf of the time delay t_{sp} can be obtained from the pdf of the spread, calculated above. Let t_r be the service time of one packet, where t_r= 1500*8/(speed of link). So, the time delay between the first packet and the last packet in the segment is equal to the number of packets in the spread multiplied by the service time of each packet. Thus, if x is the total number of packets that constitute the spread then we can write $t_{sp} = t_r * x$. Since the distribution of the time delay between the first and the last packet is the same as the distribution of the packets in the spread, we have: $P[t_{sp} = t_r * x] = P[n_K = x]$.

The end-to-end delay of the first packet in the segment consists of the propagation delay and the transmission and queueing delays at each router along the path of the segment. In our model, we have assumed that the background traffic follows a binomial distribution, i.e., for each time slot there is a probability p that a background packet arrives. Now, the combined tagged and background traffic offered to each link has to be less than the link's maximum throughput, so that the link's utilization is less than 100%. In view of this, there are no background packets queued at each router when the first packet of a segment arrives at the router. (This was also verified through extensive simulations). Therefore, the queueing delay at each link encountered by the leading packet of a segment is zero, and the end-to-end delay of the first packet is the propagation delay and sum of transmission times t_p. This leads us to the pdf of the total delay: $P[t_e = t_p + t_r * x] = t_p + P[n_K = x]$, where $P[n_K = x]$ can be determined using Eq. (3) or (4).

3.5. The client player

In HTTP live segment streaming, it is a client's responsibility to download the next segment before the previous segment is completely played out. This implies deadlines by which segments need to be encoded and be available at the video server for download. On the client's side, if a segment is not available, a deadline miss occurs, and the playback stalls. There are several segment request strategies that clients can implement. Four request strategies for live adaptive streaming are discussed and evaluated in [26]. Two of these strategies maintain a constant liveness while the other two increase the end-to-end delay after each deadline miss. The goodput of strategies with constant liveness increases as more bandwidth becomes available. The reason for this behavior is that these strategies provide a full segment duration of time for segment download. We chose the Constant Liveness Immediate Request (CoIn) strategy as it has no start-up delay and does not synchronize requests which can lead to bandwidth wastage. It maintains the liveness of one segment duration throughout the streaming session which means that a segment that becomes available at t_i at the video server is presented at t_{i+1} at the client.

A deadline miss also occurs if the download time is longer than the segment duration, t. In this case, the part of the segment downloaded after the segment playout deadline is skipped. In order to decrease the number of deadline misses, the adaptation algorithm chooses the segment quality so that the download ends at least t_s seconds before the segment deadline. Thus, a deadline miss occurs only if the download time is longer than the estimated download time plus the time safety. The minimal value of t_s is referred to as the time safety. This request strategy is illustrated in Fig. 7. A client first requests the latest segment on the server at r_0. The first segment is downloaded completely at the client at d_0 and the playout begins at t_1. The next segment is requested at r_1 and available at the client at d_1. The number of bytes that can be downloaded within the time safety increases with available bandwidth. This results in fewer deadline misses as the available bandwidth increases. In this respect, one should choose a larger time safety if more bandwidth fluctuations are expected. We can also adjust the time safety dynamically based on the observed bandwidth fluctuations.

We assume that the client makes a request immediately after $t - t_s$ seconds and that the request reaches the server before the next t-second period starts. We have used the following client rate adaptation algorithm in our model:

1. Download the first segment at the lowest bitrate
2. Determine the download time for the current segment
3. If the video segment is completely downloaded by time $t - t_s$
   a. Determine the highest bitrate so that it can be downloaded by $t - ts$ with the current available bandwidth
      i. Determine the delay per bit for the current rate (r_{curr}), i.e.,$r_{curr} = t_e/(r_{curr} * t)$
      ii. Determine the highest bitrate, r_{nxt}, for which the expected download time is the closest to $t - t_s$, i.e., $(t_e/(r_{curr} * t)) * (r_{nxt} * t) \simeq t - t_s$
   b. Send an HTTP GET request for this higher bitrate (r_{nxt})
   c. Go to step 2
4. If the video segment is not downloaded by $t - t_s$
   a. Send an HTTP GET request for the next lower bitrate for which the expected download time is closest to $t - t_s$
   b. Go to step 2

Please cite this article as: S. Tanwir, H. Perros, Modeling live adaptive streaming over HTTP, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.025

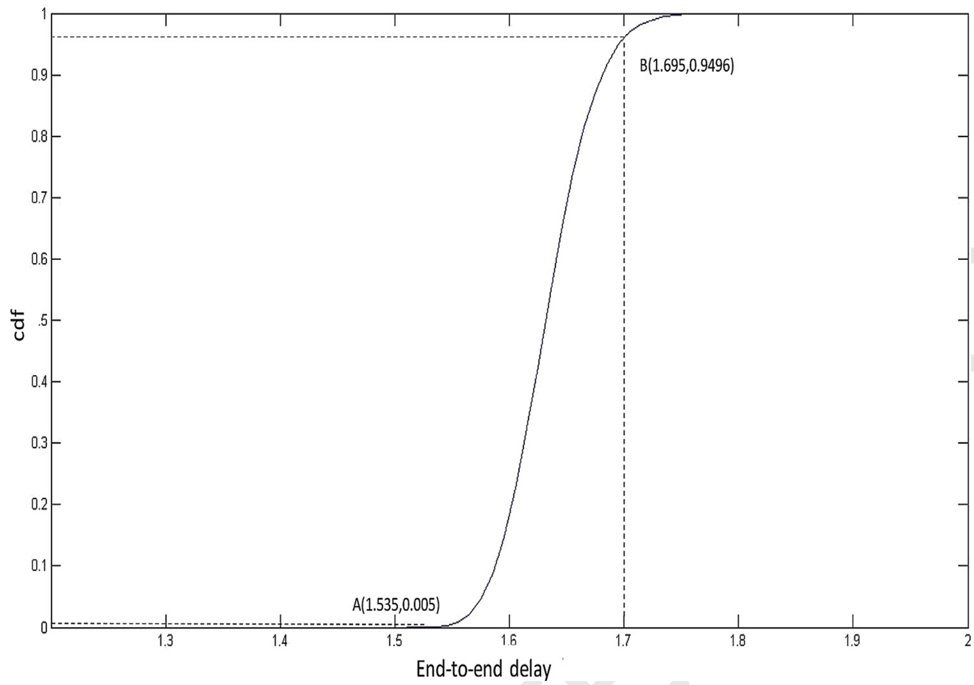

Fig. 8. CDF of the end-to-end delay for 900 Kbps bitrate.

3.6. State transition probabilities

Using the above algorithm and the cdf of the total delay for each rate, we can determine the state transition probabilities for the video source model. The total time to download a segment determines the available bandwidth which helps the client decide the bitrate to download the next segment. Therefore, we obtain the cdf from the pdf of the end-to-end delay obtained in Section 3.2. Then, we find points on the cdf beyond which the bitrate has to be changed in order to download the next segment within the deadline using the current available bandwidth.

For example, let us assume that the client can request 2 s segments with three different bitrates: 800, 900 and 1000 Kbps, and that the time safety is 0.3 s. That means $t - t_s$ is 1.7 s and the segment needs to be completely downloaded at the client by this time. Fig. 8 gives the cdf for 900 Kbps bitrate obtained assuming the queueing network shown in Fig. 3 with four WAN routers that transmit at 1.2 Gbps and one Ethernet access network node with a transmission rate of 1.2 Mbps. The background traffic is 60% of the total link capacity in the WAN and only 5% of it continues into the Ethernet access network. Each point on the cdf gives the probability that the video segment encoded at 900 Kbps will reach the client within a certain end-to-end delay (the x-axis). For example, point A indicates that the end-to-end delay will always be less than or equal to 1.535 s with a probability of 0.005. From this, we can calculate the total delay per bit, i.e, 1.535/(900,000$*$2) (since there are 900,000$*$2 bits in the 2 s segment). We can also calculate the total delay for a segment encoded at a higher bitrate assuming the same delay/bit. For example, at 1000 Kbps, the delay will be 1.7 s. Thus, A is the point beyond which if the client switches to a higher rate, the total delay taken by the new segment will be more than $t - t_s$ which is 1.7 in this case. This implies that the client only switches to a higher rate if the end-to-end delay is less than or equal to 1.535 s. This point then gives us the state transition probability of switching from 900 Kbps to 1000 Kbps.

Now, let us find the probability of switching to a rate lower than 900 Kbps. This will only happen if the total delay is greater than 1.7 s. Point B on the curve indicates that the end-to-end delay will always be less than or equal to 1.7 s with a probability of 0.9496. This means that the probability the end-to-end delay will be more than 1.7 is $1-0.9496 = 0.0504$. Also the probability that the client will request the same rate again based on the current delay is $1-0.0504-0.005 = 0.9446$.

Employing the same technique, we can calculate all rows of the transition matrix using the cdfs for different rates and for different time safety values.

4. Validation of the rate transition rates

In this section, we validate our method for calculating the rate transition probabilities of the server traffic model using simulation. (The expression of the cdf of the end-to-end delay is exact, and consequently it does not require validation). The simulation model is a discrete-time model based on the same assumptions as the analytic model described above, and it consists of a video server that generates video packets in a queueing network of 5 single-server queues and a client player that implements the rate control logic. The video server generates segments at different rates based on the requests from the client every t seconds. These segments are packetized and transmitted in the network in 1500-byte packets. The background traffic is assumed to follow a binomial distribution. A slot is equal to the amount of time it takes to transmit out a 1500 bytes packet.

The first four queues are part of the core network and we set their service rate to 1.2 Gbps. The last queue is assumed to be part of the Ethernet access network and transmits at a speed that is hundred times less than the core. All packets have the same priority. We assume that 80% of the background traffic that arrives at each queue in the core network leaves before entering the next queue, and 95% of all the background traffic leaves before entering the last queue. The background traffic is set to 60% of the link capacity.

The client implements the rate adaptation algorithm described in Section 3.5. It maintains a buffer of one video segment as we are modeling the live streaming case. The client sends the request at $t - t_s$ and we assume that it reaches the server, after a fixed

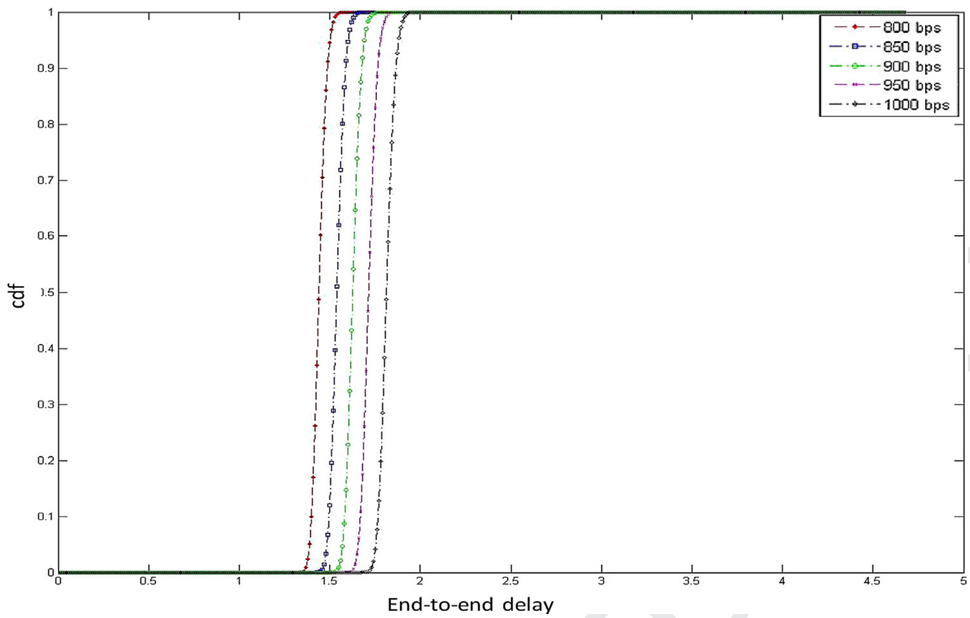

Fig. 9. The cdf of the end-to-end delay for rates 800,850,900,950 and 1000 Kbps.

delay that equals the transmission and the propagation delays before the server transmits the next segment. The simulation program was written in Matlab. The simulation model was run for a million video segment requests.

We compared the transition probabilities obtained from the simulation model with the transition probabilities calculated using the cdf of the end-to-end delay obtained from the mathematical model as explained in Section 3.6. In the simulation model, we determine the transition probabilities by counting the frequency of transitions among the bitrates requested by the client for each segment. The client requests a new bitrate after downloading each segment based on the rate control algorithm discussed in Section 3.5 This is done for a large number of segment requests. The cdfs of the end-to-end delays for a chosen set of rates are given in Fig. 9. The set of rates are determined based on the input parameters of the model, i.e., the transmission rates of the routers and the background traffic as these dictate the available bandwidth. These are the most selectable rates for given network parameters. For example, if the available bandwidth is 1 Mbps, the client will most probably select a bitrate closer to 1 Mbps instead of a very low rate, say 300 Kbps or a very high bitrate. Hence, there will be no transitions to those bitrates even if they are offered to the client.

We compared the one-step transition matrices using the Mean Squared Error (MSE), defined as:

$$MSE = \sum_{i=1}^{n}\sum_{j=1}^{n}(X_{ij} - Y_{ij})^2/size(X) \tag{5}$$

where X_{ij} are the transitions calculated using the mathematical model, Y_{ij} are the transitions determined using simulation, and $size(X)$ is the total number of elements in the matrix. The MSE of the one-step transition matrices are plotted in Fig. 10 as a function of the time safety $t - t_s$. The MSE values are very low which indicate that the transition matrices are very similar.

We conducted another set of experiments assuming faster core and access network elements. We set the transmission rate in the core network to 10 Gbps and the access network transmission to 2 Mbps. The set of video bitrates that the client can choose from are from 1650 Kbps to 1800 Kbps in increments of 500 Kbps. The cdfs of the end-to-end delay for this case are shown in Fig. 11, and the results for MSE as a function of the time safety $t - t_s$ are shown in Fig. 12.

We note that in both experiments, the MSE value is very small. Additional results were obtained for other input values including the case of the slow servers, see [27]. Based on these results, it appears that the mathematical model is very accurate and predicts the rate change probabilities very close to those obtained by simulation.

5. Applications of the model

As was seen above, our analytic model can be used to characterize the departure process of IP packets from the video server. Video traffic models are crucial in network dimensioning and resource management of IP networks. Using the proposed model, we can determine the packet arrival process for different types of networks by varying the number of nodes, link capacities, background traffic utilization and video server transmission rates. In addition, the model can be used by the video service providers iteratively to help determine the optimal video bitrates to encode the videos for given network parameters and types of clients. It also enables them to dimension the server properly to meet clients' quality of service requirements. This may include determining a maximum number of clients per output port that can be entertained simultaneously.

In the remaining of this section, we describe a new rate control algorithm which takes future decisions into consideration in order to avoid playback interruption and achieve better smoothness and quality.

5.1. Rate control algorithm

The main idea behind the algorithm is that the client estimates the available bandwidth of the network links and this information can be used to estimate the time required to download a video segment that is available at different bitrates. The client gets the information about all the bitrates, that the server offers, from the MPD file. The client constructs the cdfs for these different bitrates and then decides on the optimal rate to download the next segment. We saw in Section 3.2, that if the speed of access link is several times less than that of the WAN links (which is true in

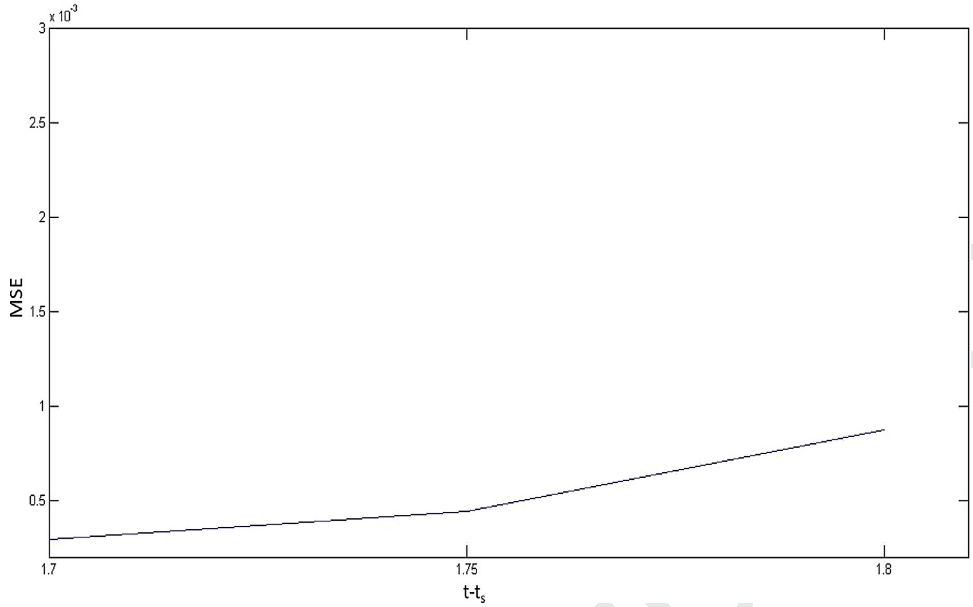

Fig. 10. Mean squared error: *Rates* 800,850,900,950,1000Kbps .

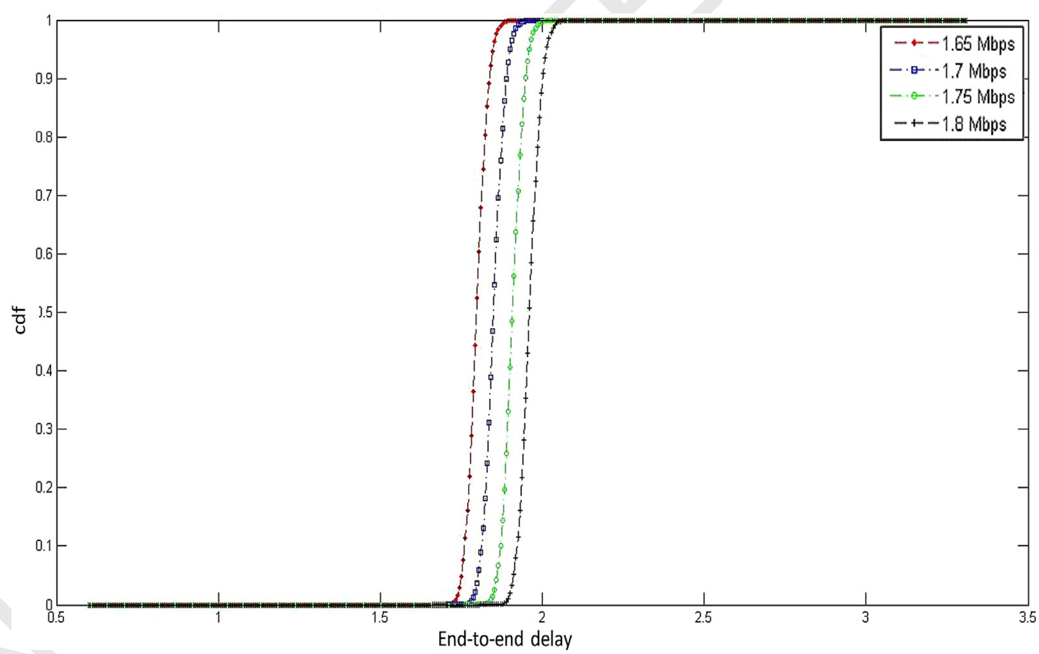

Fig. 11. The cdf of the end-to-end delay for rates 1.65, 1.7, 1.75 and 1.8 Mbps.

most cases), the spread shrinks in terms of number of packets (and slots) but takes more time to be transmitted because of the slower speed. Making use of this observation, the client can estimate the cdf of the delay by measuring the background traffic that affected the spread at the access network link only. In order to do that, the client player measures the time it took to download the complete segment. Since it knows the capacity of the link, it can also determine how much time the actual video segment data took to download out of the total time. The difference between the two gives the delay caused by the background traffic and the percentage of background traffic that affected the spread can be estimated from that. The client assumes that the background traffic arrival process is binomial and the time is slotted just as in the model.

The pdf of the number of background packets in the spread can be written as:

$$P[n_{Kb} = k] = \binom{N_s}{k} p^k (1-p)^{N_s - k} \tag{6}$$

Here p is the percentage of background packets per segment estimated by the client every t seconds. Since N_s is fixed, the pdf of the spread is same as above. From that the cdf of the spread and consequently, the cdf of the end-to-end delay can be obtained. In order to do that, the client should also measure and add the propagation delay.

We compared the pdf of the spread and the cdf of the end-to-end delay obtained by the above approximation with the ones calculated by the model. We assumed the same queueing network

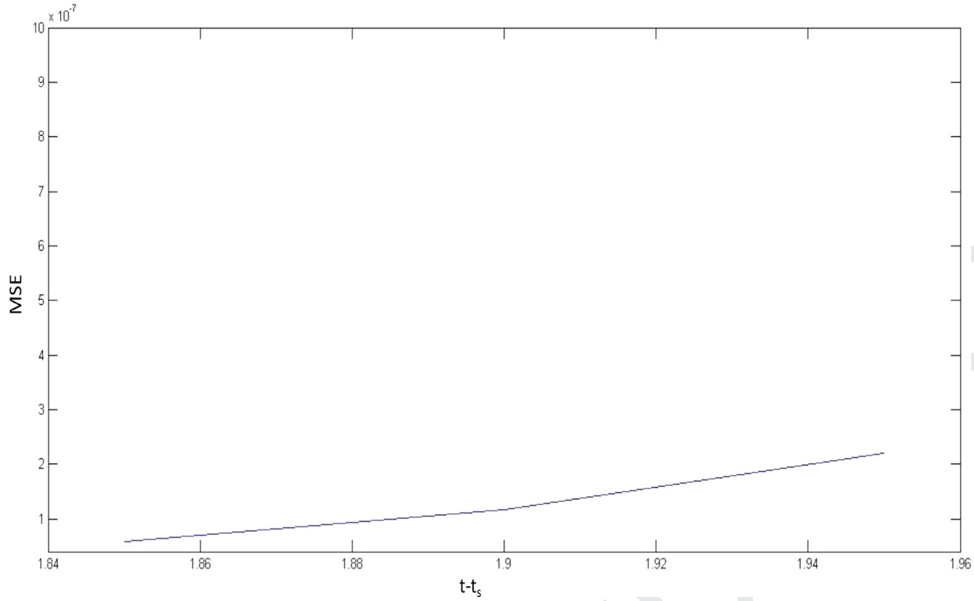

Fig. 12. Mean squared error: *Rates* 1.65, 1.7, 1.75 and 1.8 Mbps .

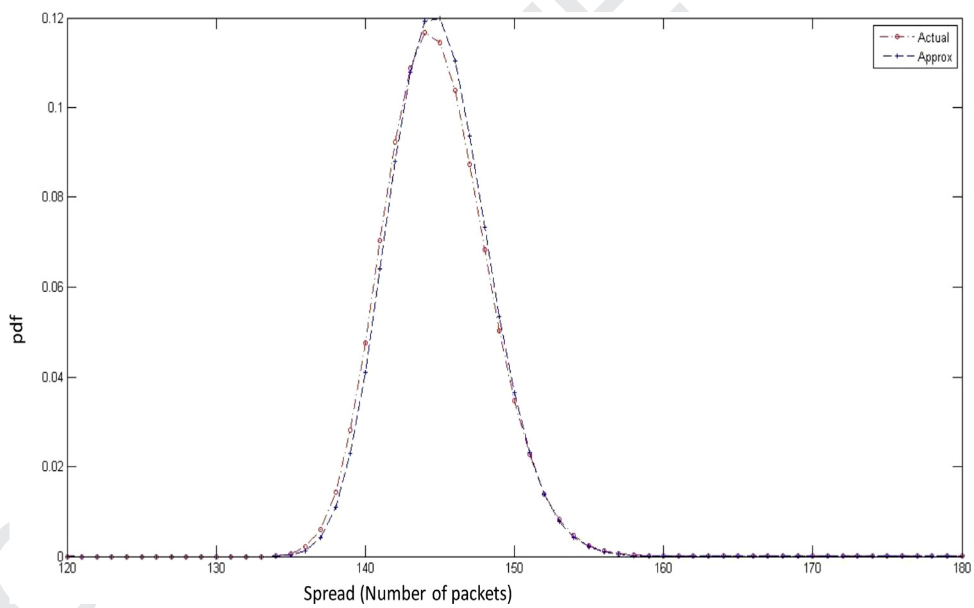

Fig. 13. The pdf of the number of packets in the spread for 800 Kbps bitrate.

model described in Section 3.2 that consists of single-server queues and a client player that implements the rate control logic. The first four queues are part of the core network and we set their service rate to 1.2 Gbps. The last queue is assumed to be part of the Ethernet access network and transmits at a speed that is hundred times less than the core. We assume that the background traffic arrives at each router in the core network and 80% of the previous background packets leave before entering the next router queue. 95% of all the background traffic leaves before entering the last queue. We assume the background traffic to be 60% of the link capacity. Only 5% of the net background traffic packets from the previous queues join the last queue and contribute to the spread. For the given input parameters, the client estimated the background packets to be 9% of the total packets in the spread at the last queue on average. Based on this percentage, we approximated the pdf of the spread and the cdf of the end-to-end delay and compared with those determined using the model. The results are shown in Figs. 13–16.

We can see that the approximated pdfs and cdfs match very well with the ones obtained using the model.

Based on the above delay estimation technique, we propose the following rate adaptation algorithm:

1. Download the first segment at the lowest bitrate
2. Determine the download time for the current segment
3. If the video segment is completely downloaded by time $t - t_s$
   a. Based on the download time of the current segment, determine the percentage of background traffic (p_{est}) that affected the spread
   b. Determine the highest bitrate so that it can be downloaded by $t - t_s$ with the current available bandwidth
      i. Determine the delay per bit for the current rate (r_{curr})

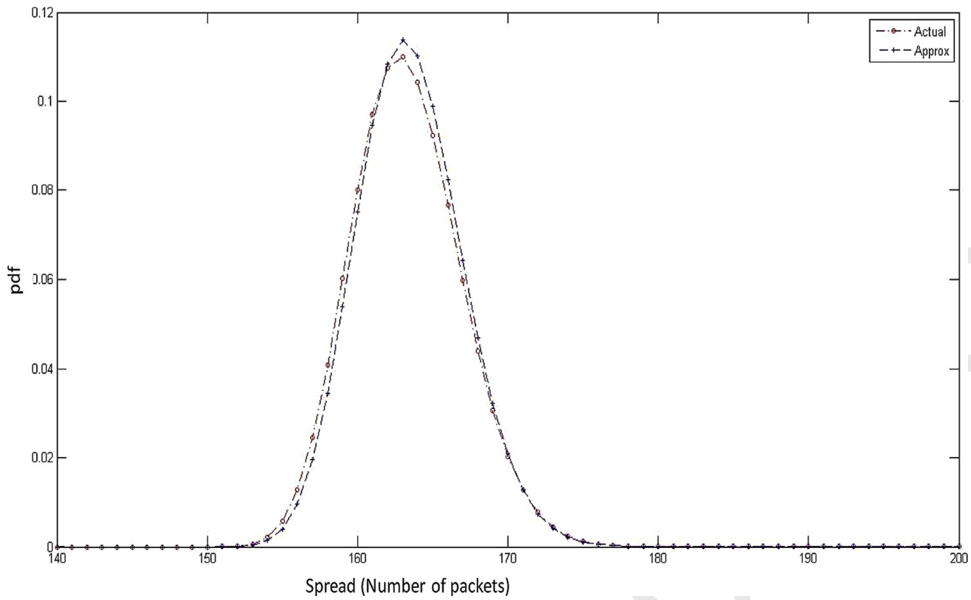

Fig. 14. The pdf of the number of packets in the spread for 900 Kbps bitrate.

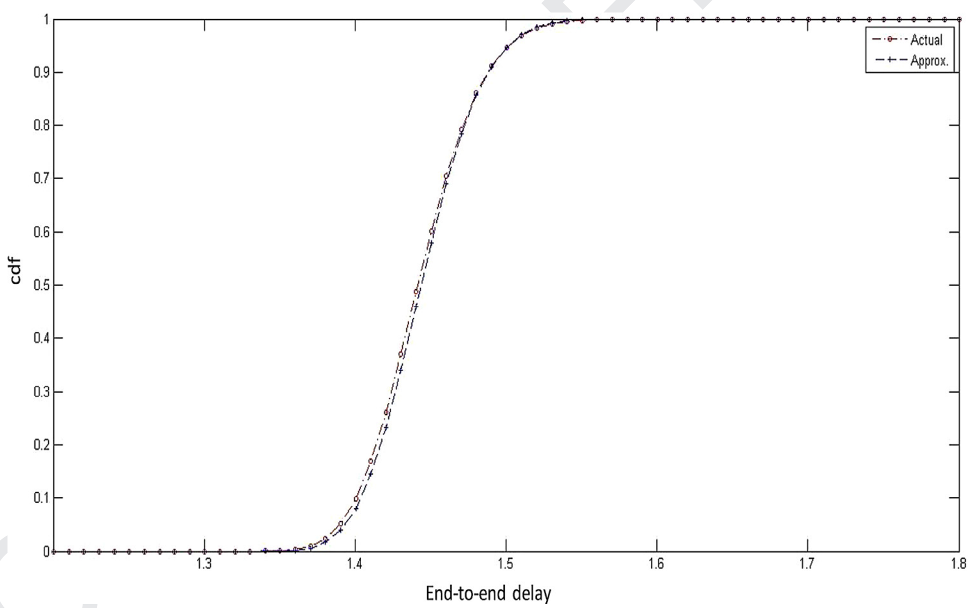

Fig. 15. The cdf of the end-to-end delay for 800 Kbps bitrate.

  ii. Determine the highest bitrate, r_{nxt}, for which the expected download time is the closest to $t - t_s$, i.e., $(t_e/(r_{curr} * t)) * (r_{nxt} * t) \simeq t - t_s$
  iii. Estimate the cdf of the end-to-end delay for r_{nxt} based on the estimated background traffic (p_{est}). Check if the 90th percentile of the delay for $r_{nxt} <= t - t_s$. If not then choose r_{nxt} as the second highest bitrate.
 c. Send an HTTP GET request for this chosen bitrate (r_{nxt})
 d. Go to step 2
4. If the video segment is not downloaded by $t - t_s$
 a. Send an HTTP GET request for the next lower bitrate for which the expected download time is closest to $t - t_s$
 b. Go to step 2

We assume that the client makes a request immediately after $t - t_s$ seconds and that the request reaches the server before the next t-second period starts. In order to smooth the rate change, we can use a moving average for the current end-to-end delay instead of the latest value. Similarly, we can use a moving average of the background traffic. The moving average can be based on last N segments, where the best value of N can be determined using simulation.

We implemented the algorithm in the simulation and compared the results with the algorithm discussed in 3.5, referred to as the "simple algorithm". We assume that the client can request 5 available bitrates at the server, i.e., 800, 850, 900, 950 and 1000 Kbps. We compared the simple algorithm with the new proposed algorithm, referred to as the "model-based algorithm", using the following metrics: the total number of rate transitions during the length of the simulation, the number of times a particular rate is selected and how often the transitions occur. In order to compare these metrics, we varied the background traffic during the simulation. This was done using a discrete time Markov-modulated Bernoulli process (MMBP) consisting of three states: low, medium

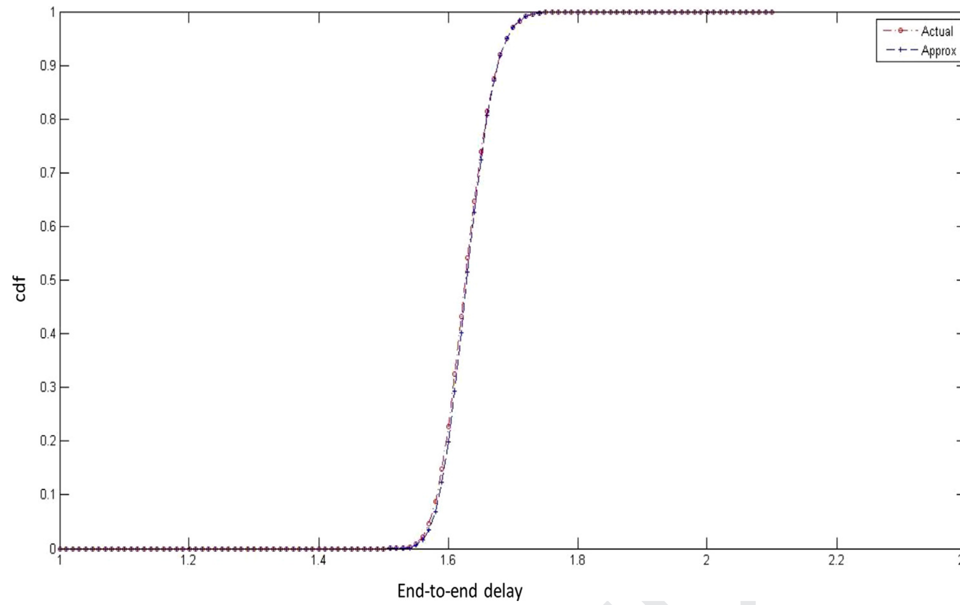

Fig. 16. The cdf of the end-to-end delay for 900 Kbps bitrate.

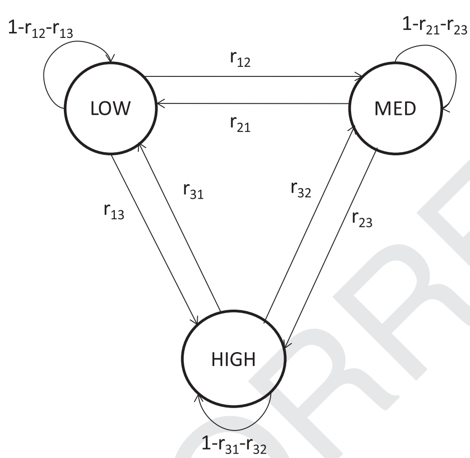

Fig. 17. Markov chain for background traffic arrival process.

and high (see Fig. 17). Within each state i, the background traffic is generated using a binomial distribution with probability p_i, set to 0.4, 0.6 and 0.7 for low, medium and high activity states respectively. The same value of p_i is used at the first four queues. At the last queue, only 5% of the background traffic joins the queue like before.

We set the state transition probabilities r_{ij} in such a way that the process spends most of the time in the medium activity state and least of the time in high activity state. Since it is a discrete-time process, time is measured in time slots. Here a slot is equal to the segment time t. During the simulation, a new state is determined every t seconds using the state transition matrix and the background traffic is generated accordingly at each queue. The state transition probabilities we used are:

$$\begin{bmatrix} 0.8 & 0.18 & 0.02 \\ 0.15 & 0.8 & 0.05 \\ 0.01 & 0.19 & 0.8 \end{bmatrix}$$

The stationary probabilities obtained after solving this matrix are:

$$\begin{bmatrix} 0.3661 \\ 0.4778 \\ 0.1561 \end{bmatrix}$$

We used a moving average for the estimated p_{est} in the algorithm. We present the results for a moving average of $N = 5$ and 10 previous segments. In Figs. 18 and 19, we present the rate transitions for the first 200 segments for both simple and the proposed model-based algorithm for two different moving average windows. We can conclude from the results that $N = 5$ is sufficient in this case. The solid line curve represents the state in which the background traffic process currently resides in. Here, state 1 is for low activity, 2 for medium activity and 3 for high activity. For this reason, we can see that when the process is in a low activity state the bitrate selected by the client is higher. Hence, the background curve fluctuates in opposite directions to the bitrate curves. We can observe from the figures that the proposed algorithm chooses the bitrate smoothly as compared to the simple algorithm described in Section 3.5. It stays in the same bitrate for longer time periods instead of choosing a higher bitrate and then choosing the same rate again like the simple client. We can see in the figure that the simple algorithm fluctuates back and forth between the 1000 Kbps and 950 Kbps bitrates but the model-based algorithm tends to choose one of these bitrates multiple times before switching to another. As discussed in [28] and [9], downloading each segment in the highest possible representation results in frequent changes of playback quality whenever the dynamics of the available throughput exhibit strong fluctuations. Therefore, it is better to choose a bitrate that will not result in too many quality fluctuations. Thus, the overall goal of the rate adaptation algorithm should be to maximize the average video quality but also to minimize the number of video quality shifts. Our proposed algorithm achieves this goal. In the case of the simple algorithm there is a transition almost every segment due to small changes in background even though the background process stays in the same state. This means that simple algorithm is more sensitive to bandwidth changes and reacts too often than necessary. However, the proposed algorithm reacts quickly similar to the simple algorithm in case of a deadline miss.

Please cite this article as: S. Tanwir, H. Perros, Modeling live adaptive streaming over HTTP, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.025

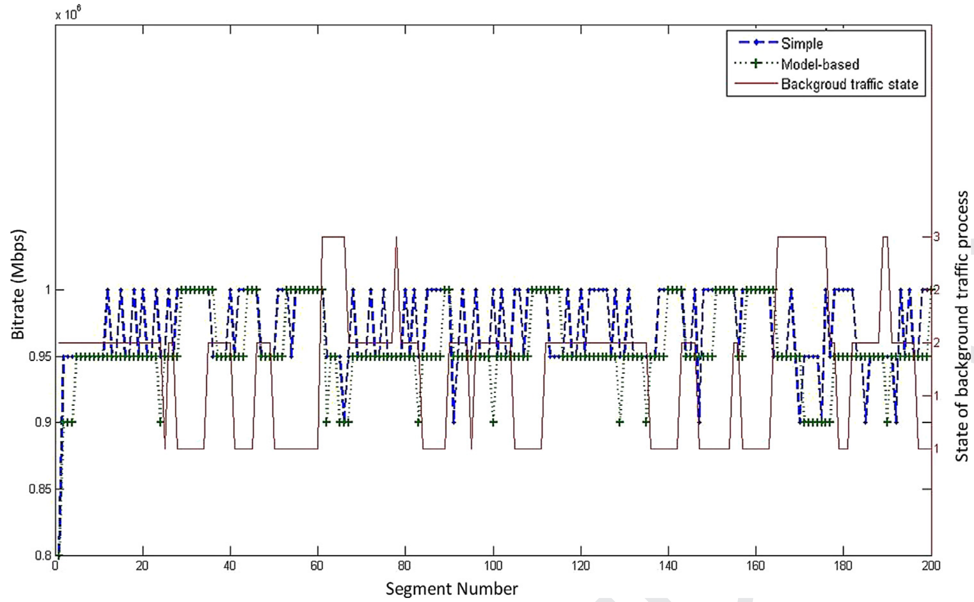

Fig. 18. Rate transitions for the simple and model-based algorithm using a moving average of the last 5 segments for p_{est}.

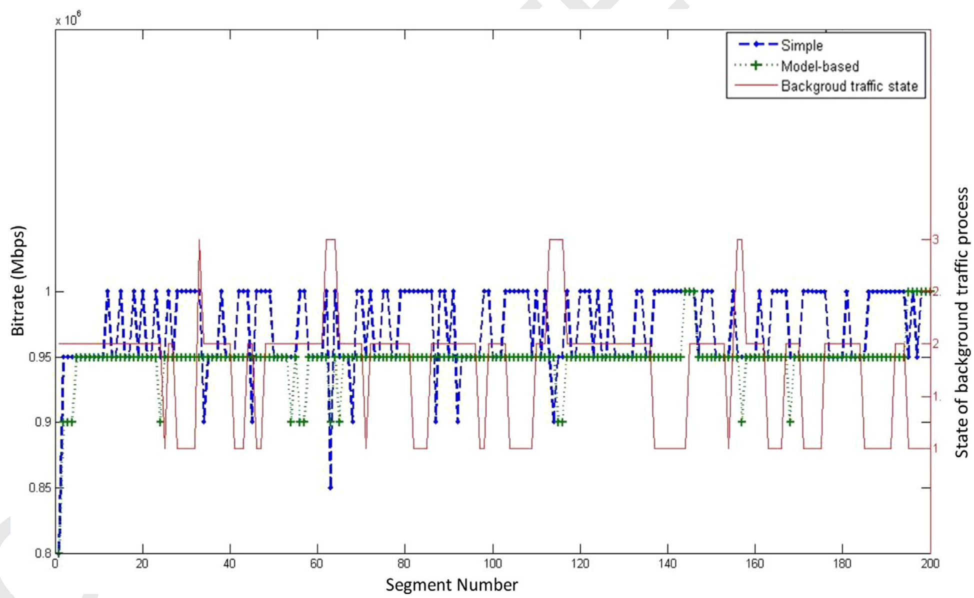

Fig. 19. Rate transitions for the simple and model-based algorithm using a moving average of the last 10 segments for p_{est}.

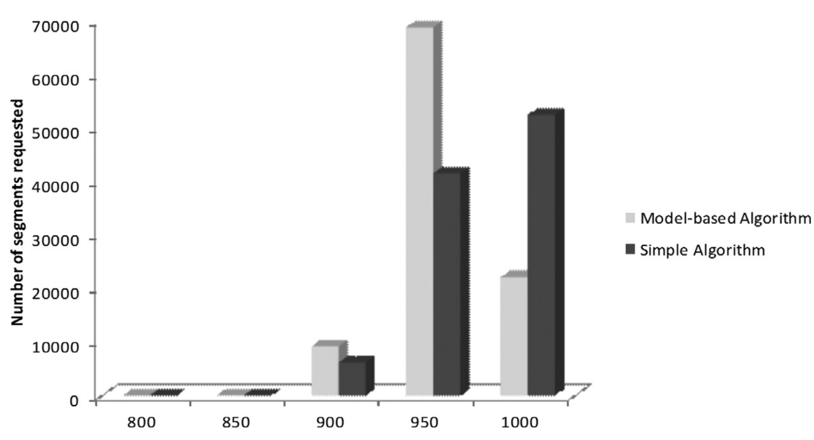

Fig. 20. Number of segments requested per bitrate for the simple vs model-based algorithm using a moving average of the last 5 segments for p_{est}.

Please cite this article as: S. Tanwir, H. Perros, Modeling live adaptive streaming over HTTP, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.025

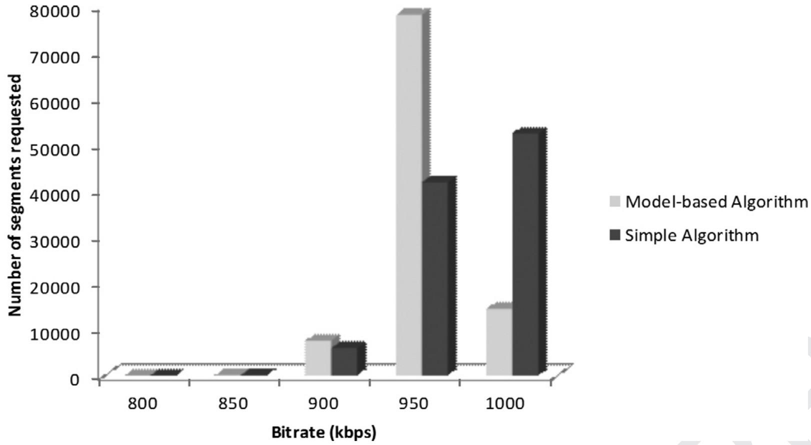

Fig. 21. Number of segments requested per bitrate for the simple vs model-based algorithm using a moving average of the last 10 segments for p_{est}.

Table 1
Total number of bitrate transitions.

Algorithm	Moving average window	Transitions
Model-based	5	15705
Simple	5	41055
Model-based	10	18884
Simple	10	41164

Next, we present the number of segments requested for each of the 5 available bitrates using both algorithms for a total of 100000 segments. We can see in Figs. 20 and 21, that the model-based algorithm requests more segments in the bitrate 950 Kbps while the simple algorithm is more aggressive and it requests more number of segments in 1000 Kbps, which results in a lot of fluctuations.

Lastly, we report the total number of bitrate transitions between the five different bitrates requested by the client in Table 1. We can see that the simple algorithm made a lot more transitions among the different bitrates as compared to the model-based algorithm. Again, this proves that the proposed model-based algorithm chooses the bitrates wisely resulting in fewer quality fluctuations and hence better quality of experience for the viewer.

6. Conclusion

Nowadays an increasing number of video applications employ adaptive streaming over HTTP, as it has several more benefits compared to classical streaming. Its offers multiple bit rates of video that enables video service providers to adapt the delivered video to the users' demands. Secondly, the video bit rate can be adapted dynamically to changing network and server/CDN conditions. Lastly, different service levels and/or pricing schemes can be offered to customers. Significant amount of work has been done on the design of rate adaptation schemes and performance comparisons, however, no one has modeled and studied the system analytically. In this paper, we proposed the first (to the best of our knowledge) analytic model for live adaptive streaming over HTTP. The model can be used to characterize the departure process of the IP packets from the video server. Also, using this model we proposed a new rate control algorithm that makes less frequent rate transitions and increases the quality of experience for the viewer.

The model is decomposed into three components, namely, the video server model, the model of the IP network, and the client video model. In the model of the IP network, we are basically interested in obtaining the distribution of the spread of a segment, and the time it takes for the leading packet of the segment to reach the client. For this, we assumed that the background arrival process is Bernoulli. In a future extension of this paper, we hope to replace it by a discrete-time bulk arrival process where the bulk size varies from one up to the total number of input ports of the router. Under this assumption the calculation of the distribution of the spread is feasible, but the calculation of the end-to-end delay becomes extremely difficult. However, this can be estimated separately for each bitrate using an extremely fast activity-based simulation reported in [29].

References

[1] Sandvine global internet phenomena report, URL https://www.sandvine.com/trends/global-internet-phenomena/. Q4
[2] Transparent end-to-end packet-switched streaming service (PSS); progressive download and dynamic adaptive streaming over http (3GP-DASH), 2011, (3GPP TS 26.247). Q5
[3] S. Akhshabi, A.C. Begen, C. Dovrolis, An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP, in: Proceedings of the MMSys, San Jose, California, USA, 2011.
[4] I. Hofmann, N. Farber, H. Fuchs, A study of network performance with application to adaptive HTTP streaming, in: Proceedings of the IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), 2011, pp. 1–6.
[5] C. Muller, S. Lederer, C. Timmerer, An evaluation of dynamic adaptive streaming over HTTP in vehicular environments, in: Proceedings of the 4th ACM Workshop on Mobile Video (MoVID), Chapel Hill, North Carolina, 2012, pp. 37–42.
[6] K. Miller, N. Corda, S. Argyropoulos, A. Raake, A. Wolisz, Optimal adaptation trajectories for block-request adaptive video streaming, in: Proceedings of the 20th International Packet Video Workshop, San Jose, CA, USA, 2013, pp. 1–8.
[7] T. Lohmar, T. Einarsson, P. Frojdh, F. Gabin, M. Kampmann, Dynamic adaptive HTTP streaming of live content, in: Proceedings of the IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM), Lucca, Italy, 2011, pp. 1–8.
[8] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hossfeld, P. Tran-Gia, A survey on quality of experience of http adaptive streaming, IEEE Commun. Surv. Tutor. 17 (1) (2015) 469–492.
[9] K. Miller, E. Quacchio, G. Gennari, A. Wolisz, Adaptation algorithm for adaptive streaming over http, in: Proceedings of the 19th International Packet Video Workshop, Munich, Germany, 2012.
[10] G. Tian, Y. Liu, Towards agile and smooth video adaptation in dynamic HTTP streaming, in: Proceedings of the International Conference on emerging Networking Experiments and Technologies (CoNEXT'12), Nice, France, 2012.
[11] A. Bokani, M. Hassan, S. Kanhere, HTTP-based adaptive streaming for mobile clients using markov decision process, in: Proceedings of the 20th International Packet Video Workshop (PV), San Jose, California, 2013, pp. 1–8.
[12] M. Xing, M. Siyuan Xiang, L. Cai, A real-time adaptive algorithm for video streaming over multiple wireless access networks, IEEE J. Selected Areas Commun. 32 (4) (2014) 795–805.
[13] A. Mansy, B.V. Steeg, M. Ammar, Sabre: a client based technique for mitigating the buffer bloat effect of adaptive video flows, in: Proceedings of the ACM Multimedia Systems Conference (MMSys), Oslo, Norway, 2013, pp. 214–225.
[14] C. Liu, I. Bouazizi, M.M.H.M. Gabbouj, Rate adaptation for dynamic adaptive streaming over http in content distribution network, Signal Process. Image Commun. 27 (4) (2012) 288–311.
[15] J. Jiang, V. Sekar, H. Zhang, Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive, in: CoNEXT'12, Nice, France, 2012.

[16] T. Schierl, Y.S. de la Fuente, R. Globisch, C. Hellge, T. Wiegand, Priority-based media delivery using SVC with RTP and HTTP streaming, Multimed. Tools Appl. 55 (2) (2011) 227–246.
[17] S. Oechsner, T. Zinner, J. Prokopetz, T. Hossfeld, Supporting scalable video codecs in a P2P video-on-demand streaming system, in: The 21st International Teletraffic Congress Specialist Seminar on Multimedia Applications - Traffic, Performance and QoE, Miyazaki, Japan, 2010.
[18] C. Sieber, T. Hossfeld, T. Zinner, P. Tran-Gia, C. Timmerer, Implementation and user-centric comparison of a novel adaptation logic for DASH with SVC, in: First IFIP/IEEE International Workshop on Quality of Experience Centric Management (QCMan), Ghent,Belgium, 2013, pp. 1318–1323.
[19] N. Bouten, M. Claeys, S. Latre, J. Famaey, W.V. Leekwijck, F.D. Turck, Deadline-based approach for improving delivery of SVC-based HTTP adaptive streaming content, in: IEEE Network Operations and Management Symposium (NOMS), Krakow, Poland, 2014, pp. 1–7.
[20] S. Akhshabi, A.C. Begen, What happens when HTTP adaptive streaming players compete for bandwidth? in: NOSSDAV12, Toronto, Ontario, Canada, 2012.
[21] T. Lohmar, T. Einarsson, P. Frojdh, F. Gabin, M. Kampmann, Interactions between HTTP adaptive streaming and TCP, in: Proceedings of the 22nd ACM Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), Toronto, ON, Canada, 2012, pp. 21–26.
[22] J. Wang, X. Hu, F. Yang, H. Luo, On modeling capacity and responsiveness of HTTP streaming, in: 2011 International Conference on Computer Science and Service System (CSSS), Nanjing, China, 2011, pp. 2358–2362.
[23] J.W. Kleinrouweler, S. Cabrero, R. van der Mei, P. Cesar, Modeling stability and bitrate of network-assisted HTTP adaptive streaming players, in: Proceedings of the 27th International Teletraffic Congress, Ghent, 2015, pp. 177–184.
[24] S. Mitra, V. Swaminathan, An optimal client buffer model for multiplexing HTTP streams, in: Proceedings of the IEEE 14th International Workshop on Multimedia Signal Processing (MMSP), Banff, AB, 2012, pp. 283–288.
[25] C. Chen, L.K. Choi, G. de Veciana, C. Caramanis, R.W. Heath Jr., A.C. Bovik, A dynamic system model of time-varying subjective quality of video streams over HTTP, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Vancouver, BC, 2013, pp. 3602–3606.
[26] T. Kupka, P. Halvorsen, C. Griwodz, Performance of on-off traffic stemming from live adaptive segmented HTTP video streaming, in: Proceedings of the 37th Annual IEEE Conference on Local Computer Networks, Clearwater, Florida, 2012, pp. 401–409.
[27] S. Tanwir, Analysis and Modeling of Variable Bitrate Video Traffic, Department of Computer Science, North Carolina State University, Raleigh, NC, 2015 Ph.D. thesis.
[28] M. Grafl, C. Timmerer, Representation switch smoothing for adaptive HTTP streaming, in: Proceedings of the 4th International Workshop on Perceptual Quality of Systems (PQS 2013), 2013, pp. 178–183.
[29] B. Anjum, H. Perros, Bandwidth estimation for video streaming under percentile delay, jitter and packet loss constraints using traces, Comput. Commun. J. 57 (2015) 73–84.

Please cite this article as: S. Tanwir, H. Perros, Modeling live adaptive streaming over HTTP, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.025