Multimed Tools Appl (2014) 72:1143–1166
DOI 10.1007/s11042-013-1424-x

Performance of HTTP video streaming under different network conditions

Arkadiusz Biernacki · Kurt Tutschku

Published online: 26 March 2013

Abstract The Internet video sharing services have been gaining importance and increasing their share in the multimedia market. In order to compete effectively and provide broadcast television with a comparable level of quality, the Internet video should fulfil stringent quality of service (QoS) constraints. However, as the Internet video is based on packet transmission, it is influenced by delays, transmission errors, data losses and bandwidth limitations which can have a devastating influence on the perceived quality of the multimedia content. There are many works which describe the impact of network impairments on the Internet video. Nevertheless, little is known about how network conditions influence the video streamed by the currently popular services such as YouTube, where video is transmitted over reliable TCP/HTTP protocols. Therefore using a network simulator, we conducted an experimental evaluation of the HTTP based video transmission analysing how the network impairments mentioned above influence the streamed video. The experiments were validated against a network emulator supplied with real network traces. As a result of this work, we can state that the buffering strategies implemented by a video player are in many cases able to mitigate unfavourable network conditions what allow to play the streamed video smoothly. The results may serve Internet Service Providers so that they could tune their network characteristics in order to match the demand from HTTP video.

Keywords Multimedia communication · Network measurements · Quality of service · Video streaming

A. Biernacki (✉)
Institute of Computer Science, Silesian University of Technology,
Akademicka 16, 44-100 Gliwice, Poland
e-mail: arkadiusz.biernacki@gmail.com

K. Tutschku
Telecommunications Systems, Communication and Computer Systems Research Laboratory (CCS), Blekinge Institute of Technology (BTH), School of Computing (COM),
371 79 Karlskrona, Sweden
e-mail: kurt.tutschku@bth.se

1 Introduction

During the last two decades, the video transmission have been considered as a demanding application that would never work satisfactorily over best-effort packet switched networks. Nevertheless, in the last few years, video-supported applications, and especially video streaming, have become quite popular. Many providers started publishing their content such as news, series, and movies on their dedicated Web sites or for this purpose used dedicated sharing services such as YouTube, Hulu in the US, Dailymotion in France, Smiley in Japan and many others. Access to the abundance of multimedia content at any time results in a drastic shift in Internet traffic statistics, which reports that the share of P2P traffic is declining primarily due to a rise in traffic from Web-based portals and video sharing services [11]. According to the YouTube Press Room, in the year 2012 the service alone had 800 million unique users each month, who viewed over 4 billion videos each month and uploaded 72 hours of video each minute. Thus the fulfilment of the rising demand for video traffic will be delegated to both content providers and ISPs (Internet Service Providers) and it is believed to be a challenging task for them.

In the above mentioned services video streaming is usually based on the HTTP and TCP, and the video player is embedded in a web browser. TCP is currently the most widely used transport protocol in the Internet but it is commonly considered to be unsuitable for multimedia streaming. The main reason lies in the TCP reliability with its retransmission mechanism which may result in undesirable transmission delays and it may violate strict time requirements for streamed live media. In this context, dealing with packet delay and loss, which can be a consequence of congestion or packet corruption, demands new solutions which will be different from the classical transmission procedures used in unreliable protocols, e.g. UDP. The HTTP and TCP are general purpose protocols and were not specifically designed or optimised for streaming media delivery. Because of their properties such as congestion control mechanisms and reliability requirement, they do not necessarily degrade the video streaming performance. By imposing proper sophisticated rate-adaptation or playback buffering, a video player is often able to adapt to the above-average throughput oscillations. Additionally, the use of the TCP, and that of the HTTP over TCP in particular, significantly facilitates the traversal of NATs and firewalls. Therefore the wide adoption of the TCP in the Internet motivated us to investigate the performance of this approach when streaming a video.

Our analysis was carried out using video streams obtained from a popular multimedia service YouTube, which primarily uses HTTP for content distribution and allows its user to watch video using a web browser. Unlike traditional VoD systems, which offer professionally-produced video content such as movies, sport events, news, reportages, YouTube videos can be uploaded by anyone with a broadband network connection. Thus, the quality and popularity of this content are not so well-controlled and predictable as in the traditional VoD systems. On the contrary, the quality of YouTube video clips varies significantly, making network optimisations for specific content unreasonable.

The quality of network connections depends on the number of statistically determined factors including latency, reliability and bandwidth. These traits are neither constant nor deterministic: in reality, they can change rapidly depending on the

local ISP network conditions, remote server behaviour, background traffic, as well as network infrastructure quality. In the case of video delivered by more traditional channels such as satellite, DVD, cable or digital TV broadcasting, data arrives at a media player with a mostly deterministic delay rate and very limited data drops. Consequently, the video delivered via such a medium usually requires little buffering space on the part of a client, whereas video delivered over the Internet is a more complex problem because there is no guarantee that the transmitted data will reach an end user with a required rate and delay. Instead, it arrives with rate and delay fluctuating during the video file transmission. Therefore buffering is of increasing importance for video streams when they are transmitted over the Internet, including Web-based streaming. Each video player usually has its own buffering and playing strategy, e.g. the Adobe Flash Player manages buffering and playing of the received content employing several algorithms [24].

In this article we study the efficiency of three client playback strategies. The study seeks to answer the question how often the player buffer runs out under different network interferences, such as packet delays, transmission errors, packet losses and throughput limitations. As a consequence, we want to investigate how these parameters influence the perceived quality of a video received by its end users. This may be useful for e.g. ISPs which may have partial influence on these characteristics and therefore may be able to tune optimally their network streamed video in order to sustain the proper quality of video transfer for their users but also simultaneously preserve their own network resources. For example, an ISP may decide to deny network resources to a new high bit rate video stream if the addition of its traffic would lead to quality degradation for the already ongoing ows of other users. Instead of gathering information from the users' applications and their viewing experience, which may be difficult in practice, the ISP may obtain information about a scale of network quality degradation by examining the above mentioned network parameters which characterise network impairments. If the newly added stream influences these parameters too heavily, and as a consequence, they become unacceptable for viewing of ongoing video streams, the ISP may consider imposing some kind of restrictions for such video stream.

To understand some of the behaviour aspects of real-time applications, we conduct a performance study using a simulation model. The simulation approach allows us to methodologically explore the behaviour of the examined system over a wide range of parameter settings, which would be a challenging task when to conduct such experiments only on a real-network. The simulation results are then confronted with the results obtained from laboratory experiments supported by real-world measurements.

2 Video distribution

2.1 Protocols

One of the classification methods of the media delivery systems is their division into systems with a feedback control and systems without any feedback control mechanism.

An example of a multimedia delivery system with the feedback control is a system using the RTSP (Real-Time Streaming Protocol). RTSP is a stateful protocol, which means that the server keeps track of a client's state from the first time the client connects to the streaming server until the time it disconnects. The client communicates its state to the server by sending commands such as play, pause or disconnect. The server begins sending the media as a steady stream of small RTP (Real-time Transport Protocol) packets. The data is sent at the media bitrate and the client buffer is filled with just a few packets before playback begins. If the client or server discover any interferences in their communications, such as increasing latency or packet drops, they can renegotiate transmission parameters e.g. the server can send the same video content but with a reduced encoding rate. The transmission is usually based on unreliable transport protocols, most commonly UDP. However, when using UDP it is often more difficult for data packets to get around firewalls and network address translators compared to HTTP/TCP protocols. The latter protocols usually operate on the same port which is used for web browsing; therefore, it is accessible via network middleboxes by default [20]. Thus, sometimes HTTP/TCP is preferred when firewalls or proxies block UDP packets, although at the expense of potentially unnecessary reliability.

Such problems are limited when employing HTTP as a media delivery protocol because firewalls and routers know to forward HTTP traffic. The protocol also does not require special proxies or caches. Additionally, HTTP is a stateless protocol; thus multimedia transmission based on it shares this feature and behaves as a system without a feedback control. Basically, if an HTTP client requests data, the remote server responds to this request by sending the demanded data; however, it stores information about neither the client nor its state. Consequently, each HTTP request is handled completely independently.

HTTP streaming may be implemented in several ways. In our work, we focus on implementation which can be described as a progressive download. The progressive download is nothing more than a transfer of a video file from a HTTP server to a client where the client may begin playback of the file before the download is complete. Contrary to the above mentioned systems with feedback control, which rarely send more than a few seconds of video content to a client in advance, HTTP streaming (web) servers progressively push the whole video content to a client and usually do not take into account how much of the data has been already sent in advance. Simultaneously, most players are capable of playing the video file while its download is still in progress. Most web-based streaming platforms, including Vimeo, MySpace and MSN Soapbox, are based on HTTP and do not have a feedback control. However, some HTTP streaming services, e.g. YouTube, implement additional flow control mechanisms at the application layer that limits the transmission rate to the same magnitude as the video bitrate [4]. Additionally, progressive download does not offer the flexibility and rich features of streaming. This has commenced the development of a new generation of HTTP-based streaming applications described as adaptive streaming over HTTP [27].

Currently, it is thought that HTTP media streaming is easier and cheaper to deploy because web streaming can use generic HTTP solutions, and does not require specialised servers at every network node. Also most of contemporary devices support HTTP in some form. HTTP relatively easily traverses middleboxes, such as firewalls and NAT devices keeping minimal state information on the server side,

which makes HTTP servers potentially more scalable than conventional streaming servers. Furthermore, a standard HTTP caching mechanism allows to move media content to an edge of the network, closer to users.

Nonetheless, the above technology also has its shortcomings. The congestion avoidance algorithm of TCP produces a saw-tooth shaped transmission rate. Furthermore, the reliability of TCP results in variable transmission delays caused by transmission errors or lost packets. If we denote the throughput of a TCP connection by r then the r is limited by the maximum segment size (MSS) and the round trip time (RTT). According to [22] and assuming that p is the probability of packet loss, the upper bound for the TCP throughput $r_{\max}$ can be calculated as:

$$r = \frac{\text{data per cycle}}{\text{time per cycle}} \leq \frac{\text{MSS}}{\text{RTT}} \frac{1}{\sqrt{p}} = r_{\max}. \tag{1}$$

As a consequence, it was commonly assumed that the video playing performance may suffer because the multimedia streaming is to some extent loss tolerant, but it is delay sensitive. Thus, the fluctuating transmission rate caused by packed delay and loss must be smoothed out by receiver-side buffering. Despite these drawbacks, currently a dominant share of multimedia traffic is being delivered using HTTP/TCP [11].

2.2 Video buffering

Most of the HTTP players are able to concurrently play and download the same file. In the simplest case, the player fills its internal buffer at the beginning of video transmission and starts video playback as soon as a minimum buffer level is achieved. While simultaneously playing and downloading the content, the amount of video data in the buffer is variable and depends mainly on the download bandwidth, video bitrate and video playing rate. When the download bandwidth is larger than the video rate the buffer grows. In the opposite case, the buffer will shrink, and if the situation lasts long enough the buffer may also run out. In such cases, the video stalls and the player waits until the buffer will be refilled again. The buffer size has to be large enough to increase the probability that the desired video quality can be achieved. However, in the current practice, there does not exist any systematic guidelines for the dimensioning of the receiver buffer and smooth play-out is usually insured through over-provisioning.

Approaching the problem from a mathematical point of view, we can assume that $G(t)$ represents the number of bytes generated at an HTTP server by time t, Fig. 1. The bytes are generated with a transmission rate limited only by the TCP throughput r in (1). The first data is generated at time 0 and sent immediately to a client. Let $A(t)$ denote the number of bytes arriving at the client by time t and $B(t)$ denote the number of bytes played by the client by time t. Since the transmission rate is limited by the generation rate at the server, we have $A(t) \leq G(t)$. Bytes arriving before their playback time are denoted as early data. At time t, the number of the early bytes is counted as $N(t) = A(t) - B(t)$. A negative value of $N(t)$ indicates that the data arrival is delayed in relation to its playback time by $-N(t)$ bytes. For a smooth video playback, it is necessary that all bytes, from which a video frame is built, arrive at a

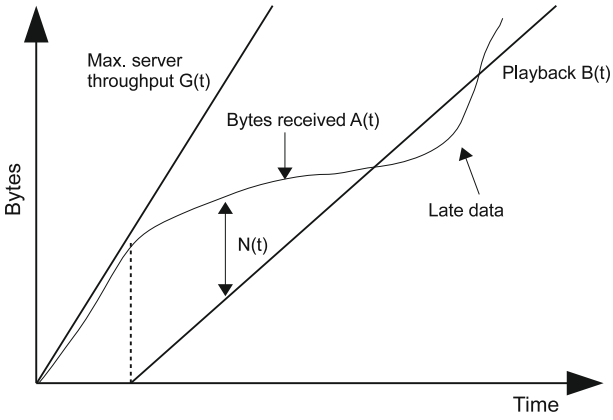


Fig. 1 Player buffer occupancy in the time function

client before the time of rendering the frame. Thus, for smooth playing, $N(t)$ should not only be a positive value, but it should also fulfil the requirement

$$N(t) \geq M(t_p), \tag{2}$$

where $M(t_p)$ is the number of bytes in the player buffer needed to play the first t_p seconds of a video.

During the streaming there can be many time periods Δt_i for which the value of $N(\Delta t_i) - M(t_p)$ is negative. In this work, we try to answer the question "what is the value of $\sum_i \Delta t_i/t$ and i, i.e. the total video stall time in a relation to video clip length and the number of stalling events for several video files transmitted from a HTTP server subjected to adverse network conditions?"

2.3 Video playing strategies

For our experiment we chose two applications capable of playing HTTP streamed video: the Adobe Flash Player and a HTML5 based player. The first, henceforth referred to as the Flash, is a proprietary browser plug-in dedicated to displaying Flash content and playing streamed Flash video. The second, henceforth referred to as the HTML5, supports videos that do not require any proprietary plug-ins, and run directly within a web browser.

When streaming with the Flash, it basically behaves like a simple HTTP player described above, i.e. it starts the video playback as soon as a minimum buffer level is achieved. However, due to the flexibility of the Flash authoring platform, the buffering functionality is additionally improved by the client-side ActionScript code. This enhancement, called a dual-threshold buffering, should make the standard buffering process more robust against bandwidth drops or other adverse network conditions, as well as it should allow the player to exploit a sudden increase of bandwidth. In the dual-threshold buffering, the playback of a video file starts when the first threshold in the buffer is filled with a given amount of data. This phase is also called initial buffering. However, the dual-threshold strategy, instead of trying to keep the buffer full to this level, tries to fill the buffer to a second, higher level. This

Fig. 2 Firefox playback (re-)start decision algorithm

```
if s_MA > v_MA then
    c ← (t_p = 20s ∨ b_T = 20s)
else
    c ← (t_p = 30s ∨ b_T = 30s)
end if
```

additional data may be useful later, if the network connection encounters temporary impairments such as bandwidth drops or fluctuations. The video play-out can start after a short pre-loading time and the excess bandwidth can be used to build a bigger reserve of data to counteract the likelihood of future network malfunctions. The details of the strategy are provided in [26].

In the case of HTML5 streaming, the playing strategy is implementation specific. The W3C HTML5 specification [18, Section 4.8] states, that in the case of autoplay "the user agent [...] will automatically begin playback of the media resource as soon as it can do so without stopping.". To approximate this difficult to fulfil condition, every implementation differs. We investigated the implementation of this specification by the Firefox because its code is an open source; therefore, the behaviour can be studied not just by observing network traces, but also by reading the sources.

The algorithm in the Firefox is summarised in Fig. 2 and Table 1. Rather than using static thresholds, it facilitates moving averages to estimate the development of the transmission rate. It does not differentiate between the initial video start-up time and intermittent buffering events. In the case when the two-second moving average s_{MA} of transmission speed is higher than the two-second moving average v_{MA} of the video bitrate, the video will be started or resumed if the player buffer initially contains at least 20 s of video data t_p (2) or the amount of time b_T spent in the non-playing buffering state playback is at least 20 s. In the opposite case, when the moving average of transmission speed is less or equal to the moving average of the video bitrate, the video will be started or resumed if the player buffer initially contains at least 30 s of video data or the amount of time spent in non-playing buffering state playback is at least 30 s. As one can notice, such implementation may require large playback buffers due to the chosen high video buffering amounts, but could also result in very few stalling events.

The parameter $M(t_p)$ defined in (2) is strictly related to an application playing strategy and can have a different value, which also can change during the time of the video play. Depending on the application used for video streaming, e.g. Flash, HTML5 or Silverlight, an HTTP server employs different streaming strategies that produce traffic patterns ranging from ON-OFF cycles to bulk TCP transfer [14].

Table 1 Variables involved in buffering decisions

Variable	Explanation
s_{MA}	Moving average of the transmission speed, set to 2 s.
v_{MA}	Moving average of the video bitrate, set to 2 s.
c	Condition upon which to start/resume playback.
t_p	Amount of video data the buffer contains (seconds).
b_T	Amount of time spent in non-playing buffering state.

3 Previous works

Many of the earlier studies of video streaming works have assumed that the underlying transport protocol is UDP (or RTP over UDP), which significantly simplified the design, evaluation and modelling of video streaming applications e.g. [7] or [32]. A major research area, related to this work, concerns the analysis and characterization of streaming services in the Internet. Many works in this field started in the last two decades of the twentieth century, and focused among others on the characterization of videos on the Web [2], the users' video access statistics [3], developing UDP-based streaming protocols, and providing mechanisms for TCP-friendliness and loss recovery e.g. [15, 25].

When we concentrate on the HTTP video, some related works aim to evaluate the performance of video streaming systems over HTTP and TCP. Cicco et al. [12] investigated the performance of the Akamai HD Network for Dynamic Streaming for Flash over HTTP. Wang et al. [30] developed discrete-time Markov models to investigate the performance of TCP for both live and stored media streaming. They showed that TCP provided good streaming performance when the achievable TCP throughput was about twice the media bitrate, with only a few seconds of delay experienced on the start.

Focusing on the YouTube, several measurement studies have been reported in literature in the last few years. These works focused on characterizing various aspects of the YouTube videos and their usage patterns. On the one hand, we have articles based on user traffic trace analysis including deep packet inspection e.g. [16, 23, 24, 34]. Their authors operated on the real world measurements obtained from e.g. ISPs' networks and they characterized video popularity, durations, size and playback bitrate, as well as usage pattern statistics such as day versus night patterns or traffic volume. Additionally, in [16] the investigation of the YouTube users' sessions statistics was presented. In [24] the authors presented a traffic characterisation of Netflix and YouTube, then they identified different streaming strategies also deriving a model for the aggregate traffic generated by these services. Plissonneau et al. in [23] described the impact of YouTube traffic on a the French regional ADSL point of presence, revealing that the YouTube video transfers are faster and larger than other large Web transfers. On the other hand, there are publications based on crawling the YouTube site for an extended period of time [1, 9, 10]. These works examined the video popularity and the users' behaviour and found that statistics such as length, access patterns, growth trend, and active life span were quite different compared to traditional video streaming applications. Furthermore, in [9] information directly available from YouTube servers was used to analyse the characteristics of videos served by the YouTube, while [10] investigated social networking in the YouTube videos. Also Abhari and Soraya in [1] investigated the YouTube popularity distribution and access patterns through the analysis of a vast amount of data collected by crawling the YouTube API. On the basis of the observations, the authors presented essential elements of the tool for emulating a variety of system workloads.

A global study of user experience for the YouTube videos using PlanetLab nodes from all over the world is performed in [21]. The results from this analysis show that on average there are about 2.5 pauses per a video file and on average 25% of the videos with pauses have the total pause time greater than 15 s. In [6] the authors

presented a characterisation of the traffic generated by YouTube and proposed a server traffic generation model.

The closest work to our work is [8] where the authors evaluated the responsiveness of adaptive HTTP algorithms under variable network conditions which included varying delays, available bandwidth, cache response times and interaction with competing traffic. The authors claimed that the performance of the streaming algorithm increased with the decrease of network delay. They experimented with providing information to the client, particularly about the achievable throughput, which balanced the noisiness of measurements and enhanced the ability of the client to accurately estimate the throughput.

4 Experiments

In order to analyse the performance of buffering strategies of HTTP video players and their robustness in varying network conditions, we prepared a controlled and isolated environment for our tests, which was presented in Fig. 3. The analyses were conducted in two steps. In the first step, we simulate the network environment between a HTTP server and its end user. The experiments are performed in the OMNeT++ simulator [28] supported by the INET framework [29] which implements the Internet protocol stacks, among others the TCP Reno which we use in our experiments. We transmitted three video files through the simulation environment and parametrised the network characteristics: delay, bit error rate, packet loss and available throughput. The output of this step is a video transmission record which includes the times and sizes of the downloaded video segments.

In the second step, the transmission record and the frame characteristics are used to feed models in a media playback emulation process in order to calculate the playback buffer fill level for every point in time during the playback. Similarly to [21], we analyse the video file assembled from received packet payload. In this process, the player forms frames from the received video data. Each of the formed frames has a specified relative time at which the frame should be played in relation to the time of

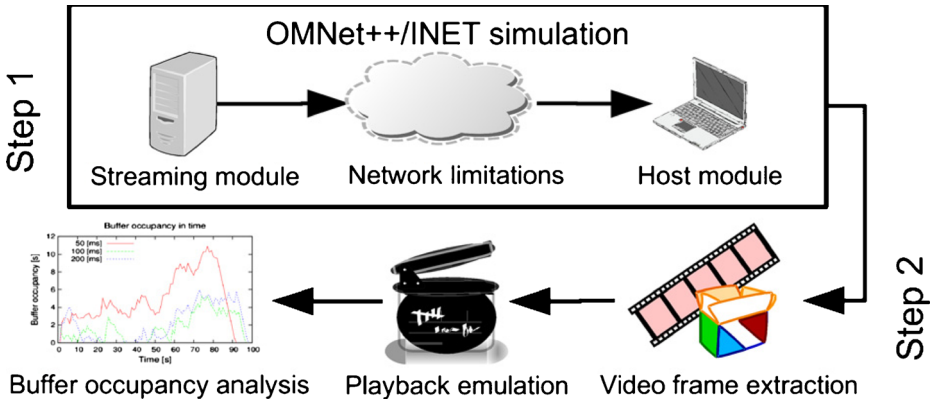


Fig. 3 Simulation environment

Table 2 Videos used during the experiments

Name	Kbps	Resolution
High bit rate (high definition)	3,177	1920 × 1080
Medium bit rate	871	854 × 480
Low bit rate	312	400 × 226

the first frame formed from the data stream. This investigation allows us to realize how much of the data is required to smoothly play each frame of the video without delays or interruptions. The models are implemented as Python scripts and imitate the behaviour of the Flash and the HTML5 video player behaviour. As an outcome, the models generate statistics of user-perceivable artifacts such as re-buffering events that occur during the playback. The above two steps were repeated five times for each video, and the computation results were made average in order to minimise measurement inaccuracies and increase the randomness of the simulation output.

For our experiment, we used a 92 s video file encoded in three different bit rates: 3177 Kbps, 871 Kbps and 312 Kbps. In the rest of the paper, we call these encodings respectively high, medium and low bit rate, see also Table 2. According to the YouTube Press Room, as of the year 2012 10 % of its video are available as high bit rate or high definition videos. We used a single video clip called "Android 3.0 Preview" downloaded from YouTube service.[1] The Flash video version of the clip was encoded using H.264/MPEG-4 Part 10 coder and delivered in the Flash Video (FLV) file format while for the HTML5 version VP8 coder and WebM file format were used. Both encoded versions have 25 frames per second.

For all experiments, the default values for network parameters are: 10 Mbps upload and download bandwidth, 2 ms delay between a host and a client, no packet loss and no error transmission. In every performed experiment, we manipulate one single parameter from the above list while other three parameters are set to their default values.

4.1 Quality measures

From the user's perspective, the key performance characteristic of a network is the quality of experience (QoE) of received multimedia content. In the context of HTTP streaming, a reliable transport protocol such as TCP is assumed, and thus video data will not be lost. However, there may be play-out interruptions caused by either bandwidth fluctuations or long delays due to retransmissions after packet loss. Furthermore, when reduced network throughput is lower than the playback rate and the buffer will drain, the video playback will pause and wait for new video data. A user expects that delays resulting from content buffering will be minimised and do not occur during normal video play. Any play-out interruptions are annoying to the end users and should be taken into account when estimating the QoE.

The QoE, based on popular methods reflecting the human perception, is a subjective assessment of multimedia quality. A user is usually not interested in performance metrics like packet loss probability or received throughput, but mainly in the current quality of the received content. However, the quality assessment is

[1] http://www.youtube.com/watch?v=hPUGNCIozp0

time-consuming and cannot be done in real time; therefore, we concentrate on these parameters which we believe impact the QoE at most. We took inspiration from objective methods of measuring the QoE proposed amongst others in [31] which for the assessment takes into account video interruptions. Other existing approaches, like [33], usually take into account full original video as a reference which can be very costly. Thus, to characterise the relationship between the application QoS and user's QoE, for our purpose, we use two measures for HTTP videos. The first measure of the application QoS takes into account relative total stalling time experienced by a user and is defined as:

$$\mathrm{SR} = \sum_i \Delta t_i / T, \tag{3}$$

where t_i are times for which $N(\Delta t_i) - M(t_p)$ defined in (2) has negative value and T denotes a total duration of the video file when played without interruptions. As the above measure is the ratio of total stalling time to the the video duration, it is desirable to minimise its value by an ISP.

The application QoS defined in (3) did not differentiate between the cases in which a user can experience one long stalling period Δt^l or several shorter stalling periods Δt^s where $\Delta t^l = \sum_i \Delta t_i^s$. Thus, in our analysis we also use a second, complementary measure which quantify the number of re-buffering events i associated with every stalling period:

$$\mathrm{RE} = \sum_i 1. \tag{4}$$

In our experiment, every video playing scenario has at least one re-buffering event which is a result of initial buffering. The initial buffering is used to accommodate throughput variability or inter-packet jitters happening at the beginning of the video play. Some streaming strategies may achieve more smoother streaming with larger initial buffering; nonetheless, it increases the start-up latency of received video content. This problem is the part of a more general topic dealing with waiting times before service consumption and interruptions during the consumption. Although, the subject has been studied for several decades in the domain of market research, in the context of Internet video services it is rather recent issue [13, 19]. The re-buffering which takes place in the middle of video playback is usually a consequence of the congestion avoidance algorithm of TCP [5].

In our analysis, we compared the SR (3) and the RE (4) of the earlier mentioned buffering algorithms: the Flash, HTML5 and simple buffering strategy (Simple). The last strategy assumes that the algorithm always starts playback as soon as any data is available in the buffer. This means that if the player is currently stalling and a complete frame becomes available in the buffer, playback will immediately restart and the frame will be shown, even if this means stopping the playback after that frame again. Such player behaviour results in the lowest required buffer space. Moreover, playing the video as soon as possible gives the fastest end. Hence the Simple strategy gives the lowest SR and an upper limit for the number of stalls occurring. Conversely, the best way to minimise the number of stalls is to wait for the entire file to be downloaded. The Simple strategy is rather not employed in practice, and in our comparison, we treat is as a kind of a benchmark for the two others strategies.

5 Results

5.1 Delays

The delay experienced by video content consists of two components: delay introduced by a network, which is the time it takes a data packet to travel from a sender to a receiver; and TCP-level delay, which is a consequence of how the TCP reacts to fluctuations in effective network throughput. The throughput fluctuations are usually the consequence of network congestion; however, they may also occur due to application-level flow control. As results of our experiments, we obtained statistics of buffer occupancy as a function of time. An exemplary trace of the buffer occupancy for one of our examined video transmission, which uses the Simple strategy, is presented in Fig. 4. We may notice that with increasing packet delay the buffer occupancy, measured as video playback time, is decreasing and re-buffering events happen more often. When the packet delay is 50 ms, there is only one stalling event at the beginning of the video transmission. However, when the delay rises to 200 ms, watching the video is rather inconvenient due to the frequent buffer under-runs. When re-buffering events occur, the video playing times lengthens and consequently the total playing time also exceeds the original video length. Further experiments concerned the statistics of buffer occupancy in the context of application QoS for which measures were defined in (3) and (4).

As is shown in Fig. 5, packet delay has a certain influence on application QoS, which is defined as the SR in (3). Gradually increasing packet delay from 0.04 s to 1.28 s causes a successive rise of the SR from less than 1 % to nearly 1000 % in the case of high bit rate video. Performance of both the Flash, Fig. 5a, and HTML5, Fig. 5b, is quite similar; however, we can notice minor differences. In the case of Flash strategy, the increase of the SR from 0.04 s up to 0.16 s causes slow grow of the SR for all three examined videos; still in the case of the HTML5 strategy the SR remains constant for the low bit rate video in the above delay range. Furthermore, the buffering strategy used in the HTML5 handles the high and medium low bit rate video slightly better when the delay is less than 0.32 s. The medium and low bit rate video have nearly identical performance for the Flash strategy when the packet delay is less than 0.32 s. In the case of HTML5, the medium and low bit rate video react similarly for the packet delay up to 0.48 s with an exception of the packet delay equal to 0.16 s. Generally, in the all cases, the delay lower than 0.16 s results in the SR below several percent, which can be considered as a safe value, acceptable by the end

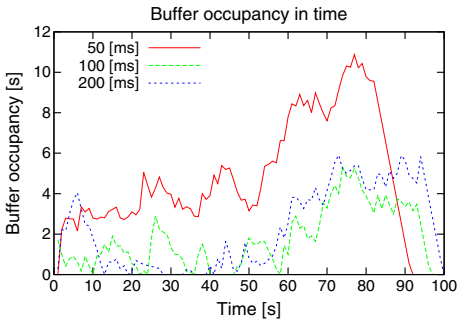
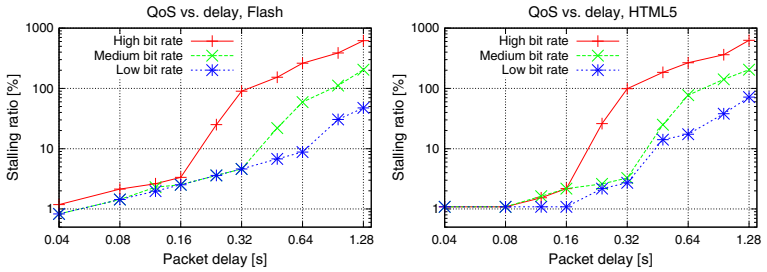


Fig. 4 Player buffer occupancy as a function of time, the Simple strategy. Simulation parameters: packet loss = 0 %, BER = 0, network throughput = 10 MBps



(a) Flash (b) HTML5

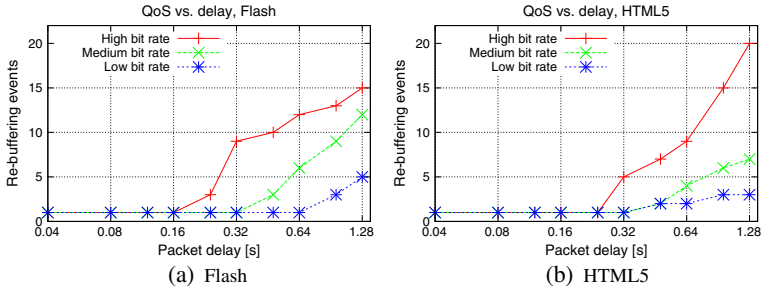
Fig. 5 The influence of network delay on the stall ratio. Simulation parameters: network throughput = 10 MBps , packet loss = 0 %, BER = 0

users. After exceeding the above mentioned delay threshold, the SR raises rapidly, especially for the high bit rate video.

When it comes to the measuring of the RE, for all of the analysed cases there is at least one re-buffering event, which exists at the beginning of the video play. Any further interruption during the video play will make the video play unfavourable for the users. According to our experiments, in order to fulfil such conditions, the maximum delay for all video types should not exceed about 0.16 s in the case of the Flash, Fig. 6a, and 0.24 s for HTML5, Fig. 6b. When we exclude the high bit rate video from our comparison, the delay upper limit can increase to 0.32 s. Both the Flash and HTML playing strategies behave more or less alike in the most cases. The HTML5 strategy is characterized by the lower RE when the packet delay is higher than 0.32 s with an exception of the high video bit rate for the packet delay equal to 1.28 s.

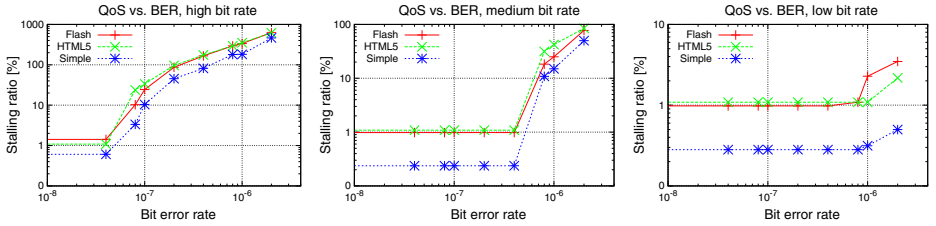
5.2 Transmission errors

Transmission errors can be measured as bit error rate (BER) which is defined as a ratio of the number of erroneous bits to the number of transmitted bits. The BER quantifies the reliability of the entire radio system including the electronics, antennas and signal path between a sender and a receiver. With a strong signal and a clear signal path, the BER value is relatively low. Nevertheless, it becomes significant



(a) Flash (b) HTML5

Fig. 6 The influence of network delay on the re-buffering events. Simulation parameters: network throughput = 10 MBps , packet loss = 0 %, BER = 0



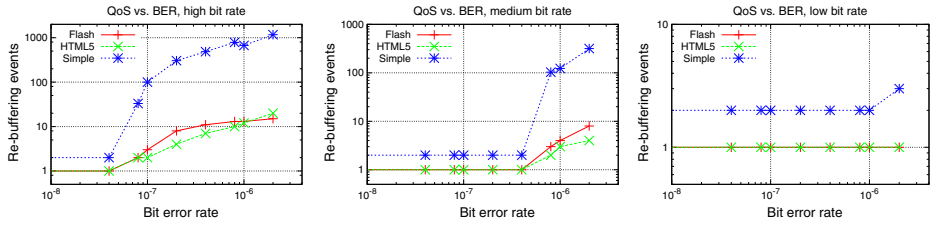
(a) High bit rate (b) Medium bit rate (c) Low bit rate

Fig. 7 The influence of transmission errors on the stalling ratio. Simulation parameters: network throughput = 10 MBps, packet loss = 0 %, packet delay = 0.02 s

when we wish to keep a sufficient signal-to-noise ratio in the presence of imperfect transmission which usually takes place when a propagation medium is a wireless link. A high level of the BER can have a devastating effect on the network leading to retransmission and as a consequence limiting the throughput, thus reducing QoS of the streamed video.

In wired networks, random BER is negligible, and congestion is the main cause of packet loss. In contrast to the fibre optical or copper wired links, wireless links use open air for transmission. This kind of medium is subjected to a whole range of uncontrollable quality-affecting factors including different weather conditions, physical obstacles, multi-path interferences and the mobility of wireless devices users. Consequently, wireless links are characterised by relatively higher BERs compared to the wired links.

In the case of streaming HTTP video, the value of the BER should not exceed 4×10^{-8} for high bit rate video, Fig. 7a. Increasing the BER from 4×10^{-8} to 10^{-6} results in dramatic incrementation of the SR value, which reaches nearly 1000 in the most extreme case. The difference in performance between the Flash and HTML5 strategies in handling the BER is nearly non-existing. Only for the BER in the range between 8×10^{-8} and 10^{-7} the difference between these two strategies are clearly visible. Both the above mentioned strategies have poorer results compared to the Simple strategy. Furthermore, after the BER exceeds 4×10^{-8}, the RE starts to rise achieving a value of two for the BER equal to 8×10^{-8} for the Flash and HTML5 strategies, Fig. 8a. When the BER rises to 4×10^{-7}, the RE for the Flash strategy stabilises in the range between about 11 and 15. For the BER values between 10^{-7}



(a) High bit rate (b) Medium bit rate (c) Low bit rate

Fig. 8 The influence of transmission errors on the re-buffering events. Simulation parameters: network throughput = 10 MBps, packet loss = 0 %, packet delay = 0.02 s

and 8×10^{-7} the RE for the HTML5 is slightly lower compared to the Flash strategy; however, with the increasing BER beyond 10^{-6} the Flash somewhat outperforms the HTML strategy.

Taking into account the medium bit rate video, we observe a high increase of the SR, Fig. 7b, and a moderate increase of the RE, Fig. 8b, when the BER is more than 4×10^{-7}. For the BER value set to 8×10^{-7}, the SR for the medium bit video rises to values between about 10 % and 25 % depending on the play-out strategy. The Flash outperforms little the HTML5 strategy when the BER is between 8×10^{-7} and 10^{-6}. The situation is reversed when it comes to measuring the RE: HTML5 strategy performs better in the above mentioned range. The HTML5 superiority may also be noticed when the BER achieves 2×10^{-6}. Traditionally, the Simple strategy shows its advantage in the case of the SR behaviour; still, in the terms of the RE measure, the Simple lags behind the Flash and HTML5 strategies.

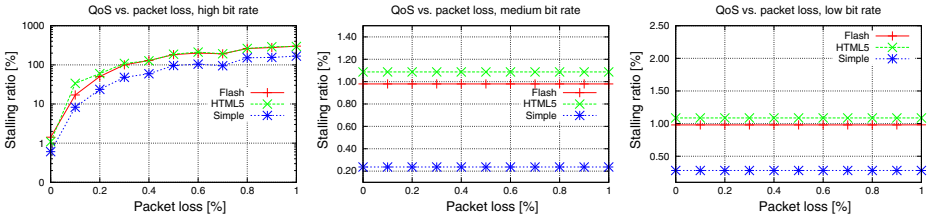
The low bit rate video is quite resilient to transmission errors and tolerates BER up to 10^{-6}. Beyond this value, the SR, Fig. 7c, and the RE, Fig. 8c, start to grow slowly. There are little differences in the SR between the Flash and HTML5 strategies for the BER higher than 10^{-6}; nevertheless, in the case of the RE, both strategies experience only a single re-buffering event, whereas the Simple strategy experiences two re-buffering events.

To sum up, we can observe differences between the Flash and HTML5 strategies for certain ranges of the BER; however, these differences tend to disappear with the increase of the BER. The Simple strategy obtains clearly worse results compared to the Flash and HTML5 strategies in terms of the RE. For the high and medium bit rate video the number of re-buffering events can be 10 to 100 times higher compared to the two other strategies. Such results show that simple playing strategies are not sufficient to mitigate some of the network flaws and impairments. Even if for simple strategies the SR is lower in comparison to the strategies used in professional video players, the other QoS measure, the number of re-buffering events, indicates that these strategies do not fulfil their role.

5.3 Packet loss

In wired networks, packet loss is caused primarily by network congestion. However, in the case of wireless infrastructure, the responsibility usually depends on the irrecoverable errors in data transmission manifested by high BER values, which lead to the corrupted packets rejected in-transit. Less common causes of packet loss are faulty networking hardware and the malfunction of network drivers or routing procedures. When transmitting HTTP video in the event of packet loss, any segments of data that have not been acknowledged are resented. Retransmitting missing data naturally causes less efficient utilisation of network throughput.

In the situation when the high bit rate video is transmitted, even 0.1 % of packets loss have a non-negligible impact on the SR for all playing strategies. As we can see in Fig. 9a, both the Flash and HTML5 strategies perform quite alike for the whole range of packet loss parameter with a small exception of the situation where the packet loss is set to 0.1 %. In this situation, the Flash slightly outperforms the HTML5 strategy. Nonetheless, the HTML5 shows its superiority over the Flash strategy, excelling in the RE measure for the whole packet loss parameter range, Fig. 10a. The Simple strategy copes slightly better with it, compared to the two others strategies



(a) High bit rate (b) Medium bit rate (c) Low bit rate

Fig. 9 The influence of packet loss on the stalling ratio. Simulation parameters: network throughput = 10 MBps, BER = 0, packet delay = 0.02 s

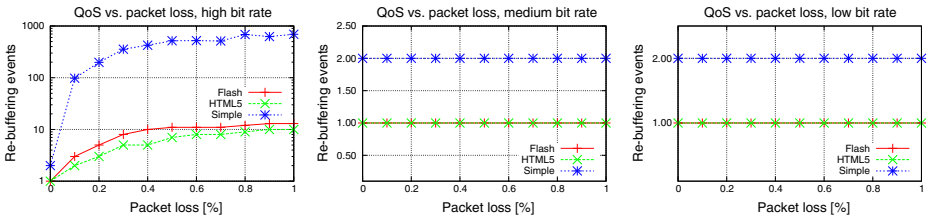
when assessing the SR; nevertheless, it is also prone to packet loss. Furthermore, a significant number of re-buffering events are experienced.

After switching the video bit rate to medium, the SR drops significantly and remains at a very low level for all the examined packet loss scope, Fig. 9b. The relative performance of the playing algorithms is similar to their performance observed in the case of the high bit rate video: the Flash and HTML5 behave quite alike. Simultaneously, the Simple strategy slightly outperform them. Considering the number of re-buffering events, the medium bit rate video is hardly affected by the packet loss, Fig. 10b. Only in the case of the Simple algorithm, there are two re-buffering events, which indicate that somewhere in the middle of the streamed video a stalling event occurred.

The quality of the low bit rate streaming is comparable with that presented for the medium bit rate video. There are minor differences between the SR for all three playing strategies, Fig. 9c. In terms of the RE, the behaviour of the playing strategies is exactly the same as in the case of the medium bit rate case: the Flash and HTML5 have one re-buffering event while the Simple strategy experiences two re-buffering events, Fig. 10c. Taking into account the robustness of the medium bit rate video against the packet loss, such results for the low bit rate video should not be surprising.

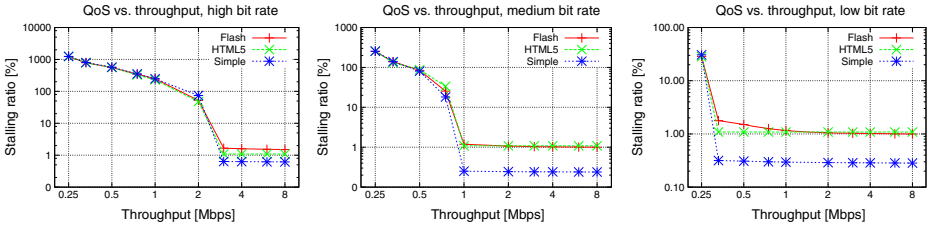
5.4 Throughput

According to [30], TCP streaming generally provides good performance when the available network bandwidth, and thus the achievable TCP throughput, is roughly



(a) High bit rate (b) Medium bit rate (c) Low bit rate

Fig. 10 The influence of packet loss on re-buffering events. Simulation parameters: network throughput = 10 MBps, BER = 0, packet delay = 0.02 s



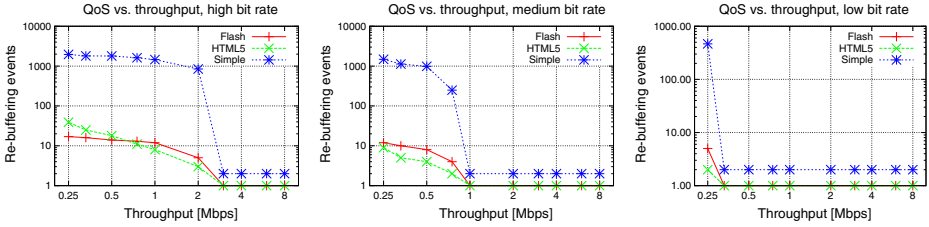
(a) High bit rate (b) Medium bit rate (c) Low bit rate

Fig. 11 The influence of network throughput on the stalling ratio. Simulation parameters: packet loss = 0 %, BER = 0, packet delay = 0.02 s

twice the video bitrate, with only a few seconds of start-up delay. However, our study shows that the statement made in [30] is alleviated, and the throughput requirements are reduced. The lower requirements for the network throughput are the results of improvements in TCP, the larger video player buffers and the better playing strategies.

Figures 11 and 12 show the dependency between the SR, RE and the throughput ranging from 256 Kbps up to 8 Mbps. Regarding the high bit rate video, the SR is quite high when the throughput is below 3 Mbps. Above this threshold, the SR drops below 5 %, which we can consider as an acceptable value, Fig. 11a. All the three strategies behave nearly identically for the whole throughput parameter scope. Furthermore, the throughput reduction beyond 3 Mbps causes an increase of the RE for all the buffering strategies, Fig. 12a. When the throughput is between about 0.75 Mbps and 2 Mbps, the HTML5 behaves better compared to the Flash strategy; however, after the throughput drops below 0.5 Mbps, the Flash strategy dominates.

The minimum network throughput for the medium medium bit rate should be no less than 1 Mbps. Below this threshold, the SR grows significantly for all the three strategies. The HTML5 strategy has the best performance when taking into account re-buffering events, Fig. 12b. The Flash playing strategy is not much worse with only several re-buffering events more compared to the HTML5 strategy. The Simple strategy is the least resilient to network throughput. The number of re-buffering events is dramatically high when the network throughput drops below 1 Mbps. The



(a) High bit rate (b) Medium bit rate (c) Low bit rate

Fig. 12 The influence of network throughput on re-buffering events. Simulation parameters: packet loss = 0 %, BER = 0, packet delay = 0.02 s

situation looks much better when the throughput rises to 1 Mbps and above, where a user of the Simple strategy experiences only two re-buffering events.

The experiments reveal that for the low bit rate video, the bandwidth requirements are much lower; therefore the link throughput above 512 Kbps should be sufficient for smooth video playing, Fig. 11c. Increasing the throughput beyond this value significantly improves neither the SR nor the RE, Fig. 12c.

Generally, taking into account the video encoding rates listed in Table 2, the streaming behaviour presented in Figs. 11 and 12 is logical and justified. The network throughput below the video encoding rate should be insufficient for smooth video streaming. Furthermore, the theoretical thresholds mentioned above are increased by network protocols. In all cases, the network throughput exceeding 15% of the video encoding rate is sufficient for a smooth video play-out which shows that the requirements regarding the network throughput presented in [30] are not as high. Nonetheless, that all the three playing strategies cannot satisfactorily handle even a small limitation in the network throughput and the initial buffering strategy implemented by the Flash and HTML5 may be not sufficient in this situation.

5.5 Results validation

To validate our results, parts of the simulation environment were replaced by laboratory equipment. Instead of sending the video files between a server and a user over TCP protocol stack implemented in OMNeT++/INET, the transmission is done through a network emulation node using the built-in Linux Kernel module *netem* [17], Fig. 13. The module provides functionality for testing protocols by emulating the properties of wide area networks being capable of altering the network QoS parameters such as delay distributions or a packet loss rate. In our validation, we focused on the scenarios with parametrised packet delay and losses. We do not take into account BER and throughput influence because such relation will be hard to obtain in the network emulator. For the analysis, we selected high and medium bit rated video played by the Flash and HTML5 strategies since for these clips the

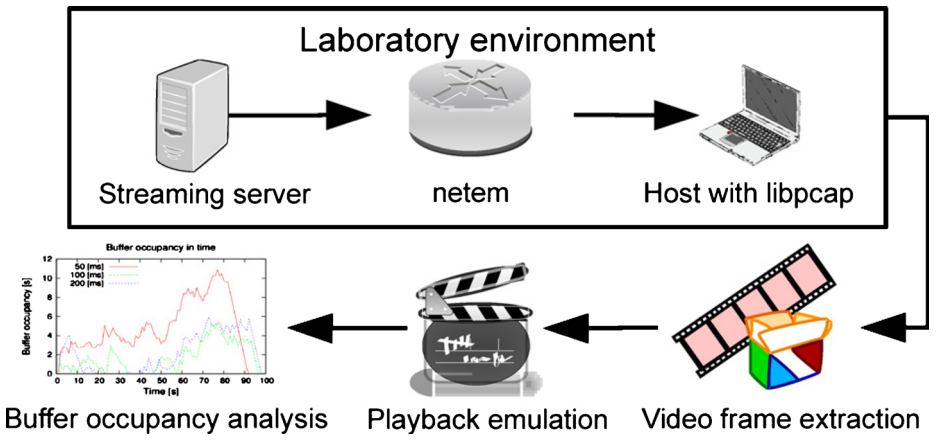
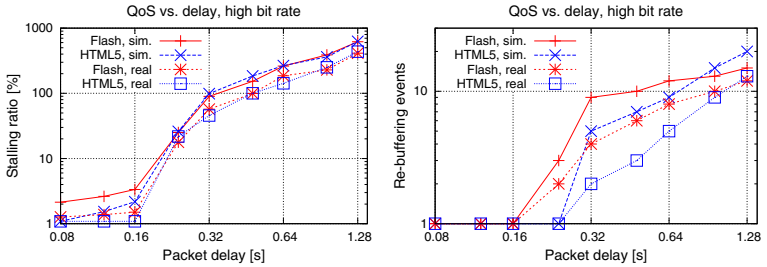


Fig. 13 Laboratory environment used to validate the simulation results

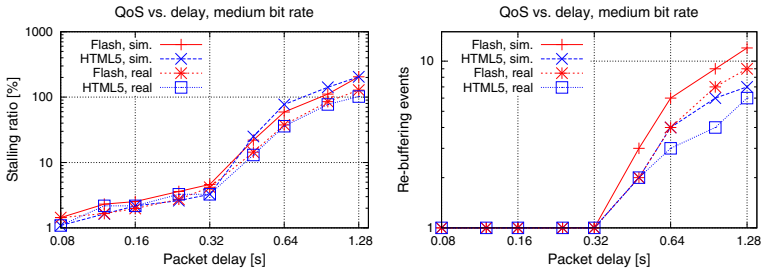


(a) Stalling ratio (b) Re-buffering events

Fig. 14 Simulation validation of high bit rate video subjected to network delay. Experiment parameters: packet loss = 0 %, BER = 0, network throughput = 10 MBps

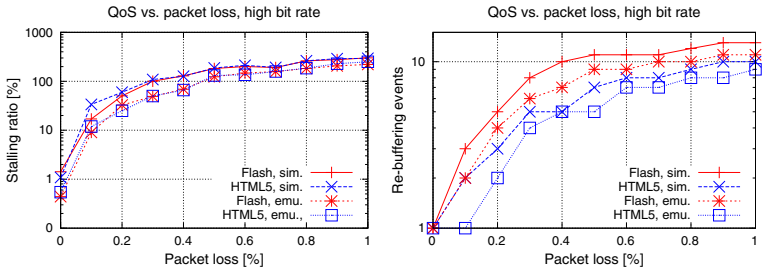
SR and RE measures generate a large range of values. Therefore such comparison allows us to validate our model for a broad spectrum of produced results. In order to eliminate potential observations that are numerically distant from the rest of the obtained data, simulated and emulated scenarios were repeated five times, and the results generated in every repetition were finally averaged.

When streaming the high bit rate video, the match between results generated by the simulation and these generated in the laboratory environment are moderately decent for both examined playing strategies, Fig. 14a. Furthermore, taking into account a relative percentage deviation, the produced outputs of the simulation and emulation are similar for most measurement points. The simulation model overestimates the SR for about 20 %; the lowest error is achieved for packet delay equal to 0.24 s while the highest deviations are observed for low values of this parameter. Similar dependency between the simulation and emulation results can be noticed when we consider the re-buffering events: the simulation model indicates more stalling events than the emulation, Fig. 14b. The highest difference can be observed for packet delay equal to 0.32 s for both Flash and HTML5 strategies. With the increasing packet delay, the gap between the simulation end emulation for the Flash strategy diminishes whereas it is rather constant in the case of the HTML5



(a) Stalling ratio (b) Re-buffering events

Fig. 15 Simulation validation of medium bit rate video subjected to network delay. Experiment parameters: packet loss = 0 %, BER = 0, network throughput = 10 MBps



(a) Stalling ratio (b) Re-buffering events

Fig. 16 Simulation validation of high bit rate video subjected to packet loss. Experiment parameters: packet delay = 0.02 s, BER = 0, network throughput = 10 MBps

strategy. Contrary to the SR comparison study, presented in Fig. 14a, the best match between the simulator and emulator is achieved for low values of the packet loss.

The contrast between the simulation and emulation in an estimation of the SR for the medium bit rate video, Fig. 15a, is lower than for the high video bit rate, Fig. 14a. The difference in the assessment is quite satisfactory for low values of packet delay. Especially for the HTML5 strategy performance, the differences are observable when the value of the packet delay is higher than 0.32 s. The largest deviation between the simulation and emulation may be noticed for the packet delay equal to 0.64 s. As for the re-buffering events, Fig. 15b, for the packet delay up to 0.32 s, the simulation and emulation models record only a single stalling event and, as a result, there is a perfect match between the models. Once the packet delay exceeds the value mentioned above, the match between the results deteriorates and the simulation records one or two additional re-buffering events in comparison to the emulation.

Taking into account the validation of the simulation model subjected to packet losses, the simulation of both the Flash and HTML5 strategies overestimate the output generated by the emulator for about 10–15 % for the packet loss equal or above 0.5 %, Fig. 16a. The worst match is observed for lower values of the examined parameter where the difference between the simulation and emulation results reaches dozens of percent. There is also difference of one or two re-buffering events between the outputs of simulation and emulation, Fig. 16b. This difference is not directly related to the packet loss values.

6 Conclusions

In the paper, we analysed how network limitations, manifested as latency, transmission errors, packet loss and bandwidth caps, impact the quality of video streamed through HTTP and TCP. For this purpose, we conducted a simulated evaluation of video transmissions with a different bit rate denoted as high, medium and low bit rate videos. We examined the QoS of an end user application, which was measured as a function of playback buffer occupancy, investigating how long and how often the video playback is stalled due to the buffer under-runs. The end user application

implemented three different buffering algorithms and playing strategies: Flash, HTML5 and Simple. In all the examined cases, we assumed that the simulated environment is isolated and there are no other network interferences (e.g. background traffic) except these introduced and controlled by us.

In our analysis, we found minor differences in the buffering algorithms and some inefficiencies in each of them. The performance of both Flash and HTML5 playing strategies is quite similar for different video bit rate. Our research confirms also that real-time application performance over TCP may not be as unsatisfactory as it has been commonly believed [30].

In order to provide the satisfactory level of a high quality video for the end users, the packet delay should be lower than 0.16 s regardless of the buffering strategy. For medium and low quality video this requirement may be loosened and the delay should not exceed respectively around 0.36 s and 0.48 s. Reducing the packet delay below the above thresholds results in a relatively short stalling time and no stalling events during the play of the video except the beginning of the play.

When transmitting high quality video, the bit transmission error rate (BER) should not exceed 4×10^{-8}. High quality video is also susceptible to packet loss: even 0.1 % packet loss greatly increases relative buffering times. In the case of medium and low quality video, the BER should remain below 10^{-6}, and even 1 % packet loss does not degrade the video playback.

For smooth transmission of high quality video, network connection throughput should have the bandwidth of at least 3 Mbps. Comfortable watching of medium quality video can be achieved for connection throughput starting from 1 Mbps, and 512 Kbps in the case of low quality video transmission. Generally, in all cases, the network throughput should exceed the video encoding rate for about 15 % what indicates that the throughput requirements are lower than it was previously expected.

Employing admission control mechanism, ISPs may have a partial influence on the above characteristics and therefore may be able to tune the quality of video transfer and influence its users' satisfaction. For example, an ISP may decide to deny network resources to a new high bit rate video stream if the addition of its traffic would lead to quality degradation for the already ongoing ows of other users. For this purpose, the ISP may monitor its network characteristics and compare it with the parameters thresholds proposed above.

The results obtained from the simulation were validated against laboratory experiments, which involved streaming and measuring transmission characteristics of real video files from a web server to a video client. We compared the stalling ratio and a number of re-buffering events for the simulated video transmissions and the transmissions of videos in laboratory environment, while manipulating with packets delay and packet loss in the network. The comparison showed that the performed simulation decently matched the modelled system; however, the produced results by the simulation are usually too conservative.

Acknowledgements The research was partially supported by the National Science Centre (Poland) under grant DEC-2011/01/D/ST6/06995.

References

1. Abhari A, Soraya M (2010) Workload generation for YouTube. Multimedia Tools Appl 46(1):91–118
2. Acharya S, Smith BC (1997) Experiment to characterize videos stored on the web. In: Proceedings of SPIE, vol 3310, p 166
3. Acharya S, Smith B, Parnes P (2000) Characterizing user access to videos on the world wide web. In: PROC SPIE INT SOC OPT ENG, vol 3969, pp 130–141
4. Alcock S, Nelson R (2011) Application flow control in YouTube video streams. ACM SIGCOMM Comput Commun Rev 41(2):24–30
5. Allman M, Paxson V, Stevens W (1999) TCP congestion control. http://www.hjp.at/doc/rfc/rfc5681.html. Accessed 18 Jan 2013
6. Ameigeiras P, Ramos-Munoz JJ, Navarro-Ortiz J, Lopez-Soler J (2012) Analysis and modelling of YouTube traffic. Trans Emerg Telecommun Technol 23(4):360–377. doi:10.1002/ett.2546/abstract
7. Apostolopoulos JG, Tan W, Wee SJ (2002) Video streaming: concepts, algorithms, and systems. HP Laboratories, report HPL-2002-260
8. Benno S, Esteban JO, Rimac I (2011) Adaptive streaming: the network HAS to help. Bell Labs Tech J 16(2):101–114
9. Cha M, Kwak H, Rodriguez P, Ahn YY, Moon S (2007) I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. In: Proceedings of the 7th ACM SIGCOMM conference on internet measurement, pp 1–14
10. Cheng X, Dale C, Liu J (2008) Statistics and social network of youtube videos. In: 16th international workshop on quality of service, 2008. IWQoS 2008, pp 229–238
11. {Cisco} (2011) Global mobile data traffic forecast update, 2010–2015. Cisco Visual Networking Index. White Paper
12. De Cicco L, Mascolo S (2010) An experimental investigation of the akamai adaptive video streaming. In: HCI in work and learning, life and leisure, pp 447–464
13. De Pessemier T, De Moor K, Joseph W, De Marez L, Martens L (2012) Quantifying the influence of rebuffering interruptions on the user's quality of experience during mobile video watching. IEEE Trans Broadcast PP(99):1–5. doi:10.1109/TBC.2012.2220231
14. Finamore A, Mellia M, Munafo M, Torres R, Rao SR (2011) YouTube everywhere: impact of device and infrastructure synergies on user experience. Tech. rep
15. Floyd S, Handley M, Padhye J, Widmer J (2000) Equation-based congestion control for unicast applications. SIGCOMM Comput Commun Rev 30(4):43–56. doi:10.1145/347057.347397
16. Gill P, Arlitt M, Li Z, Mahanti A (2007) Youtube traffic characterization: a view from the edge. In: Proceedings of the 7th ACM SIGCOMM conference on internet measurement, pp 15–28
17. Hemminger S (2005) Network emulation with NetEm. In: Linux conf au, pp 18–23. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.67.1687&rep=rep1&type=pdf
18. Hickson I, Hyatt D (2011) HTML5: a vocabulary and associated APIs for HTML and XHTML. W3C Working Draft edition
19. Hofeld T, Egger S, Schatz R, Fiedler M, Masuch K, Lorentzen C (2012) Initial delay vs. interruptions: between the devil and the deep blue sea. In: 2012 fourth international workshop on quality of multimedia experience (QoMEX), pp 1–6. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6263849
20. Khlifi H, Gregoire JC, Phillips J (2006) VoIP and NAT/firewalls: issues, traversal techniques, and a real-world solution. IEEE Commun Mag 44(7):93–99. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1668388
21. Krishnappa DK, Khemmarat S, Zink M (2011) Planet YouTube: global, measurement-based performance analysis of viewer;'s experience watching user generated videos. In: 2011 IEEE 36th conference on local computer networks (LCN), pp 948–956
22. Mathis M, Semke J, Mahdavi J, Ott T (1997) The macroscopic behavior of the TCP congestion avoidance algorithm. ACM SIGCOMM Comput Commun Rev 27(3):67–82
23. Plissonneau L, En-Najjary T, Urvoy-Keller G (2008) Revisiting web traffic from a DSL provider perspective: the case of YouTube. In: Proc. of the 19th ITC specialist seminar
24. Rao A, Lim Y, Barakat C, Legout A, Towsley D, Dabbous W (2011) Network characteristics of video streaming traffic. In: CoNEXT, Tokyo, Japan
25. Rejaie R, Handley M, Estrin D (1999) Quality adaptation for congestion controlled video playback over the internet. ACM SIGCOMM Comput Commun Rev 29:189–200

26. Sinergia P (2006) Implementing a dual-threshold buffering strategy in flash media server. http://www.adobe.com/devnet/adobe-media-server/articles/fms_dual_buffering.html
27. Stockhammer T (2011) Dynamic adaptive streaming over HTTP: standards and design principles. In: Proceedings of the second annual ACM conference on multimedia systems, pp 133–144. http://dl.acm.org/citation.cfm?id=1943552.1943572
28. Varga A, Hornig R (2008) An overview of the OMNeT++ simulation environment. In: Proceedings of the 1st international conference on simulation tools and techniques for communications, networks and systems & workshops. Article No. 60. ICST (Institute for Computer Sciences, Social Informatics and Telecommunications Engineering). Accessed 18 Jan 2013
29. Varga A, Hornig R. INET. inet.omnetpp.org
30. Wang B, Kurose J, Shenoy P, Towsley D (2008) Multimedia streaming via TCP: an analytic performance study. ACM Trans Multimedia Comput Commun Appl (TOMCCAP) 4(2):1–22
31. Watanabe K, Okamoto J, Kurita T (2007) Objective video quality assessment method for evaluating effects of freeze distortion in arbitrary video scenes. In: Proceedings of SPIE, vol 6494, p 64940P
32. Wu D, Hou YT, Zhu W, Zhang YQ, Peha JM (2001) Streaming video over the internet: approaches and directions. IEEE Trans Circuits Syst Video Technol 11(3):282–300
33. You J, Reiter U, Hannuksela M, Gabbouj M, Perkis A (2010) Perceptual-based quality assessment for audio visual services: a survey. Signal Process Image Commun 25(7):482–501
34. Zink M, Suh K, Gu Y, Kurose J (2009) Characteristics of youtube network traffic at a campus network-measurements, models, and implications. Comput Netw 53(4):501–514

Arkadiusz Biernacki received the M.Sc. and Ph.D degree in Computer Science from the Silesian University of Technology, Poland, in 2002 and Ph.D. 2007 respectively. From 2007 he is an Assistant Professor at the Silesian University of Technology. From 2010 he has been collaborating with Chair of "Future Communication" (endowed by Telekom Austria) at the University of Vienna. His research interests focus on network traffic modelling and computer system simulations.

Kurt Tutschku is the Professor for Telecommunications Systems at the Blekinge Institute of Technology, Karlskrona, Sweden. Before that, he had Chair of "Future Communication" (endowed by Telekom Austria) at the University of Vienna (until February 2013) and worked from February 2008 to July 2008 as an Expert Researcher at the NICT (National Institute for Information and Communication Technology, Japan). Furthermore, Kurt Tutschku was from August 1999 to December 2007 an Assistant Professor at the Department of Distributed Systems of University of Wuerzburg where he led the department's group on Future Network Architectures and Network Management. Kurt Tutschku has received a doctoral degree in Computer Science from University of Wuerzburg in 1999 and completed his habilitation at the University of Wuerzburg in 2008. His main research interest include future generation communication networks, Quality-of-Experience, and the modelling and performance evaluation of future network control mechanisms and P2P overlay networks.