QDASH: A QoE-aware DASH system

Ricky K. P. Mok[§], Xiapu Luo[§], Edmond W. W. Chan[†‡], and Rocky K. C. Chang[§]
Department of Computing[§]
The Hong Kong Polytechnic University
{cskpmok|csxluo|csrchang}@comp.polyu.edu.hk
Corporate Research Department[†]
Huawei Research, China
edmond.chan@huawei.com

ABSTRACT

Dynamic Adaptation Streaming over HTTP (DASH) enhances the Quality of Experience (QoE) for users by automatically switching quality levels according to network conditions. Various adaptation schemes have been proposed to select the most suitable quality level during video playback. Adaptation schemes are currently based on the measured TCP throughput received by the video player. Although video buffer can mitigate throughput fluctuations, it does not take into account the effect of the *transition* of quality levels on the QoE.

In this paper, we propose a QoE-aware DASH system (or QDASH) to improve the user-perceived quality of video watching. We integrate available bandwidth measurement into the video data probes with a measurement proxy architecture. We have found that our available bandwidth measurement method facilitates the selection of video quality levels. Moreover, we assess the QoE of the quality transitions by carrying out subjective experiments. Our results show that users prefer a gradual quality change between the best and worst quality levels, instead of an abrupt switching. Hence, we propose a QoE-aware quality adaptation algorithm for DASH based on our findings. Finally, we integrate both network measurement and the QoE-aware quality adaptation into a comprehensive DASH system.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations – *Network monitoring*; H.1.2 [**Models and Principles**]: User/Machine Systems – *Human factors*; H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems – *Evaluation/methodology*

General Terms

Design, Measurement, Human Factors

[‡]This work was partially done when the author worked at The Hong Kong Polytechnic University.

MMSys'12, February 22-24, 2012, Chapel Hill, North Carolina, USA.

Keywords

QoE, Dynamic Adaptive Streaming over HTTP, Available Bandwidth, Quality Adaptation, H.264/AVC

1. INTRODUCTION

Billions of videos are watched every day via HTTP streaming [2]. By using standard HTTP to transmit video data, HTTP streaming can easily traverse firewalls and NAT devices. Furthermore, the progressive download technology [10] enables users to watch incompletely downloaded video clips. However, users have increasing expectations of video streaming, from improved usability to better Quality of Experience (QoE). Dynamic Adaptation Streaming over HTTP (DASH) is proposed to improve the user perceived quality [34].

Before using DASH, video clips are first encoded into multiple quality levels by adjusting the video bit rate, resolution, frame rate, or audio bit rate. The resulting set of clips is referred to as an MBR (Multiple Bit Rate) video. The quality level is the main control option in both HTTP streaming and DASH. In the classic HTTP streaming, users can only choose the quality levels manually. This is clearly insufficient for providing the best QoE, as a user's choice may not be optimal and may not adapt to fast-changing network conditions, particularly in wireless and mobile networks [32].

DASH, as an extension of the classic HTTP streaming, is able to automatically switch the video quality level according to the current network conditions. Moreover, the video clips are logically divided into small fragments according to the size of GOPs (Group of Pictures). The fragment size is usually between two to ten seconds of video [13]. Before downloading each fragment, the video player decides the most suitable quality level based on the historical TCP throughput of the last or last few fragments.

The video player usually chooses a quality level that has a lower bit rate than the measured throughput, so that the download rate is higher than the playback rate. This avoids rebuffering caused by buffer starvation [13, 26]. The behavior of different video player implementations, however, is not the same. For example, the Netflix player is more forceful in attempting to provide higher quality levels, while Microsoft Smooth Streaming is on the more conservative side [11].

Measuring the throughput on the client side is relatively straighforward. An estimate can be obtained by dividing the number of downloaded bytes by the download time. However, averaging techniques, such as exponential average or weighted average, are often applied to the measurement results to mitigate short-term fluctuations caused by the lower

layers. If the averaging period is too long, the result cannot correctly reflect on the current network conditions. Hence, wrong decision on choosing the quality levels could be made.

On the other hand, playing at a higher quality level does not always result in a higher QoE. We should also consider the transition of quality levels. This is especially relevant to DASH, as the quality changes with the network conditions. Normal users do not usually watch the same video more than once within a short period of time. Therefore, they have no reference to judge the video quality. However, they can identify the spatial artifacts by comparing past experiences of other videos they have watched. Previous studies [17, 31, 36] have shown that users show little appreciation for quality improvements, while they heavily criticize quality degradation. Therefore, users may prefer a lower initial quality rather than a sudden decline in video quality when the network throughput drops.

This paper proposes a comprehensive and QoE-aware DASH system, named QDASH. This system aims at providing a practical solution to improve the QoE for the current DASH systems. The video service providers do not need to re-encode existing video clips and do not require to install extra softwares in the server. QDASH composes of two modules – QDASH-abw and QDASH-qoe.

QDASH-abw is a novel probing methodology to detect the highest quality level the current network conditions can support. This module is implemented in a measurement proxy. The measurement proxy is placed in front of the video server and manipulates the packets in video data flows. QDASH-abw measures the available bandwidth by RTT variations [21]. However, we speed up the convergence of available bandwidth estimates by only considering a few discrete sending rates which match to the bit rate of video quality levels. we also exploit the packet sending sequence to increase the number of RTT samples.

With the available bandwidth estimates, QDASH-qoe is responsible for helping clients to select the most suitable video quality level. To understand the perceived quality of video quality adaptation, we carry out subjective assessments. The assessment simulate different value of buffer size and switching approaches. Our results show that inserting intermediate levels provides a better QoE than directly switching to the target quality level. With this finding, we propose a QoE-aware adaption algorithm to utilize the video buffer and select a suitable quality levels.

The rest of this paper organizes as follow. Section 2 describes the background of DASH and existing quality adaption algorithms. Section 3 gives an overview on the QDASH system. Section 4 illustrates the methodology of QDASH-abw. Section 5 examines the QoE impacts of quality adaption and proposes the QDASH-qoe. Section 6 highlights some related works, and finally we conclude the paper in section 7.

2. BACKGROUND

2.1 DASH

The principle of DASH is to encode video clips into multiple bit rates or quality levels, by varying the resolution, picture, audio quality, etc. After the encoding, the video clips are logically divided into *fragments* by generating meta data in the file header. The meta data maps the video fragment position to video time according to the size of GOPs (Group of Pictures) [3]. When the appropriate modules are installed on the web server, the video fragments can be randomly accessed by clients.

Figure 1 shows a typical deployment of a DASH system. A client attempts to watch a video that has been encoded into three quality levels A, B, and C. The video server first delivers a DASH-capable video player to the client's web browser. While the client is watching the video stream, the quality adaptation algorithm in the video player continuously updates the recommended quality level according to the network throughput measured by the video player. The video player then requests the respective video fragments via standard `HTTP GET` requests. By continuously playing consecutive fragments, the video player can reassemble the fragments (the first three fragments with level A, followed by four fragments with level B and the last three fragments with level C) into a seamless video playback. Adobe Dynamic Streaming [9], Microsoft Smooth Streaming [28] and Apple HTTP Live Streaming [12] are popular implementations of DASH systems.

2.2 Quality Adaptation Algorithms

The quality adaptation algorithm is the core component of DASH. It controls the quality level of fragments to be downloaded. Algorithm 1 shows the quality adaptation algorithm used by the Adobe's Open Source Media Framework (OSMF) [5]. Before downloading each fragment, the video player will run this procedure once to determine the most suitable quality level. The procedure actually relies on the historical network throughput by recording the time taken to download the last video fragment.

Besides quality level switching, the video buffer is often used to mitigate possible spatial or temporal artifacts caused by sudden change in network conditions. The video player can request fragments prior than the actual play time. Even the network throughput shrinks below the bit rate of current quality level, the quality level can still be maintained for a short period of time by consuming the buffer. At the same time, the adaptation algorithm suggests for a new quality levels to avoid the occurrence of rebuffering events.

Recently proposed quality adaptation algorithms for DASH, such as [19, 26], select a quality level that is as close as possible to the network throughput. A commonly used strategy to swap between quality levels is AIMD (Additive Increase and Multiplicative Decrease) [20, 17]. The MD component produces a sharp degradation in playback quality. This strategy under-utilizes the buffer to provide intermediate quality levels to enhance the QoE. Hence, we propose a buffer-aware strategy to overcome this shortcoming.

3. OVERVIEW OF QDASH

QDASH consists of two building blocks – QDASH-abw and QDASH-qoe. QDASH-abw measures the network available bandwidth, and QDASH-qoe determines the video quality levels. These two modules can be integrated into existing DASH systems, while the modifications to the systems are kept to minimum.

QDASH is designed for streaming H.264/AVC video clips, and aims at immediate deployment to current systems. The main shortcoming for using H.264/AVC in DASH is that the storage overhead is large for multiple copies of video for different bit rates. To reduce the overhead, researchers recently propose employing H.264/SVC ,which encodes video clip

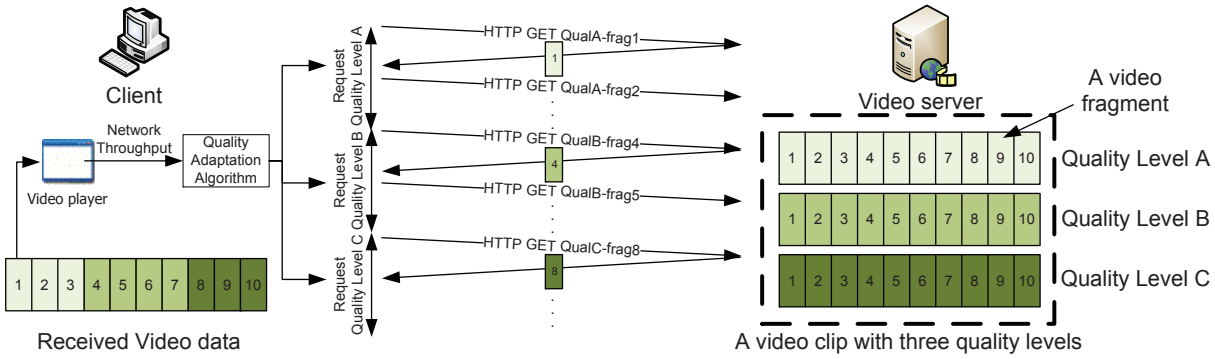


Figure 1: A typical DASH system.

Algorithm 1 Quality adaptation algorithm in OSMF [5]

$t_{lastfrag}$: Time of downloading the last fragment
l_{cur}: Current quality level
l_{nxt}: Proposed quality level
l_{min}: Lowest quality level
l_{max}: Highest quality level
$b(l)$: Bit rate of quality level l
$r_{download} \leftarrow \theta / t_{lastfrag}$
if $r_{download} < 1$ **then**
  if $l_{cur} > l_{min}$ **then**
    if $r_{download} < (b(l_{cur}-1)/b(l_{cur}))$ **then**
      $l_{nxt} \leftarrow l_{min}$
    else
      $l_{nxt} \leftarrow l_{cur} - 1$
    end if
  end if
else
  if $l_{cur} < l_{max}$ **then**
    if $r_{download} \geq (b(l_{cur}-1)/b(l_{cur}))$ **then**
      repeat
        $l_{nxt} \leftarrow l_{nxt} + 1$
      until $(l_{nxt} = l_{max})$ **or** $(r_{download} < (b(l_{nxt}+1)/b(l_{cur})))$
    end if
  end if
end if

into enhancement layers [33], to improve the efficiency. However, millions of existing video clips have already been encoded into multiple bit rates with H.264/AVC codec. They only need to insert meta data in order to enable DASH, while H.264/SVC solution requires video re-encoding which is computationally expensive. Therefore, the proposed architecture is targeted for DASH systems using H.264/AVC.

Figure 2 shows the overall QDASH's architecture. A measurement proxy, which is equipped with the QDASH-abw, is directly connected to the video server. Therefore, all the network traffic to the video server has to pass through the measurement proxy. It inspects video data flows and shapes the packet sending rate according to the bit rates of video quality levels and sends packets according to the QDASH-abw probing method (c.f. Section 4). The proxy is an IP-less device, which does not require an IP address and is transparent to both clients and servers.

Employing the measurement proxy paradigm reduces the overhead of measurement. The proxy manipulates real video data packets to perform *inline measurement*. Extra probing packets, such as the RTT test in Akamai's video streaming [18], are not required. Furthermore, by coupling the measurement flow with the video data flow, we avoid using additional measurement flows, which may not go through the same path as the data flow because of load balancers on the path. Hence, measured performance can be obtained with higher accuracy and less overhead.

With the measurement proxy, the server-side is able to measure their clients without installing additional softwares or requiring root privilege at either side for lower levels' information. The measurement tasks are offloaded to the measurement proxy and do not induce additional loading to the video server. The server-side only needs to modify the video player at the application level. The client-side also does not need to install softwares, such as libpcap [7], to cooperate with the measurement [25].

At the client-side, we proposed a QoE-aware quality adaptation algorithm – QDASH-qoe (c.f. Section 5). We first perform subjective experiments to evaluate the effect of quality switching on perceive quality under the same network scenario. We found that introducing an intermediate level could smooth out the abrupt changes in video quality and hence improve the QoE. QDASH-qoe makes use of this founding to adjust the video quality. It can be implemented in the video player delivered to the clients' browser. On the other hand, QDASH enabled video player to establish a lightweight flow to receive updates on the measurement results measured by QDASH-abw. At the same time, this flow can also report application level events or user-viewing activities to help inferring the QoE [30].

The video delivery procedure for QDASH is described as follow. Clients first sends an HTTP request to the video server to initiate the video streaming. The video server transmits the HTTP responses with the requested video fragments. Then, the measurement proxy hijacks the data flows and transmits the data packets according to QDASH-abw probing methodology. Hence, the RTT can be measured by using the acknowledgement packets triggered by the re-ordered data packets. At the same time, clients can start playing the video once the video buffer is full. Clients also connect to the measurement proxy for the latest measurement results and choose the most suitable quality level with QDASH-qoe.

4. MEASURING AVAILABLE BANDWIDTH FOR DASH

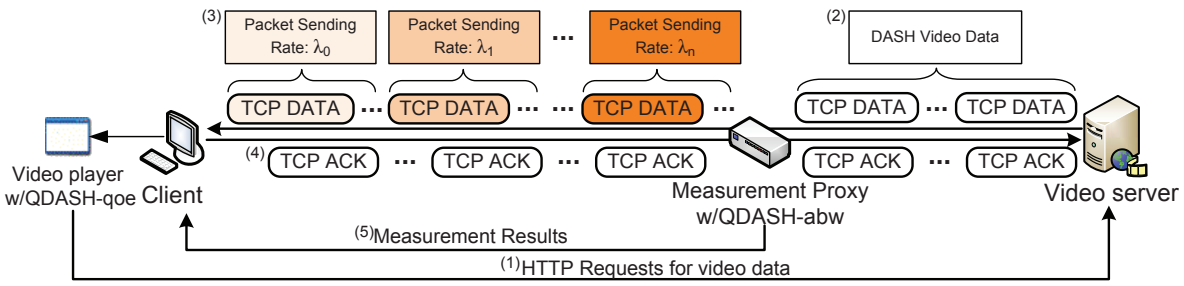


Figure 2: The overall QDASH architecture.

Instead of simple throughput measurement, we employ available bandwidth measurement to assist clients to suitably select the video quality levels. Available bandwidth measurement methods, such as Pathload [21], aim at acquiring accurate estimates by varying the packet size or the packet sending rate. However, these tools need tens of second to obtain one estimate, and this time is too long for DASH to adjust the quality.

In fact, video quality levels have a limited number of values. To determine whether the current available bandwidth is higher than any of the video quality levels, a high-resolution estimate is not required. Instead, timely updates of estimates are more important for correctly selecting or withdrawing the quality levels. We therefore propose a quantization approach to reduce the number of probes and speed up the convergence of results.

4.1 Assumptions

We make the following assumptions for the available bandwidth measurement methodology:

1. The average bit rates of video quality levels are known.
2. The available bandwidth between the server and the client is always higher than or equal to the least video bit rate (the lowest video quality level).
3. The client has sufficient computational power to render all the video quality levels.

These assumptions are realistic. The service providers are often responsible for encoding video clips. They can therefore easily obtain the video quality level information from the encoding profiles. Our probing methodology (c.f. Section 4.2) does not test the available bandwidth lower than the least video bit rate. Moreover, DASH cannot help improve the scenario if the client consistently has low available bandwidth. In this paper, we are only interested in the change in quality levels for adapting to the network conditions, instead of other factors, such as clients' computational power and system loading. Therefore, we assume all the clients can smoothly render the highest quality video clips.

4.2 QDASH-abw Probing Methodology

The basic idea of our measurement method comes from the observation that if the packet sending rate λ is higher than the available bandwidth A, the mean and the variance of the packets' round-trip times (RTT) will be larger than the values obtained when the packet sending rate is less than the available bandwidth [35]. Although the basic idea is simple, there are three challenging issues to be tackled:

1. How to generate the probing packets for measurement?
2. How to collect the measurement samples?
3. How to determine the acceptable sending rate?

These issues are closely related to the design of QDASH and we will elaborate on our solutions below.

To the best of our knowledge, existing tools for available bandwidth measurement use customized probing packets. Part of the available bandwidth is consumed for the measurement. In contrast, we propose using inline measurement that employs the media data packets directly to determine whether a certain sending rate is supported by the current available bandwidth. The rational behind such design is two-fold. First, measurement results from an additional TCP connection may not be accurate because the new TCP connection may not go through the same network path as the media data because of the load balancing. Second, the additional measurement packets will waste the bandwidth.

A probing round is defined as a sequence of packets sending at the same rate from the measurement proxy. For each probing round, the server side (measurement proxy) elicits a packet train with K pairs of W-byte probe packets at a sending rate λ. We denote the TCP data packets sent from the video server by $Sa|b$ and the response packets from the client by $Ca|b$. a and b are the data segments' sequence number and acknowledgement number, respectively. Instead of showing the exact TCP sequence and acknowledgement number, we simply use $a = 1, 2, 3, ...$ to label server's TCP data segments and $b = 1', 2', 3', ...$ client's data segments.

Figure 3 illustrates an example of two probing rounds with sending rates (kbps) λ_0 and λ_1, and the available bandwidth is between λ_0 and λ_1. We assume that the TCP connection between the client and the video server has been established, and the server receives an `HTTP GET` request from the client for the first video fragment. The response packets do not send to the client directly, and are intercepted by the measurement proxy. At the slow start phase of TCP connection, the sending window in the video server is small. The number of on-the-flight respond packets is not enough for a probing round. Hence, the measurement proxy needs to send pure TCP ACKs to the video server for fetching more video data packets. In each probing round, the measurement proxy buffers $2K$ data packets, $S1, ..., S2K$. Then, it sends a TCP ACK with zero receive window to suppress the server from sending out more video data packets and overflow the proxy. After a new probing round starts, the proxy re-opens the sending window of the video server by sending a duplicated TCP ACK with non-zero window size. In our implementation, we set the window size to two times of the Maximum Segment Size (MSS).

Existing tools for available bandwidth measurement usually adopt binary search technique for the sake of determin-

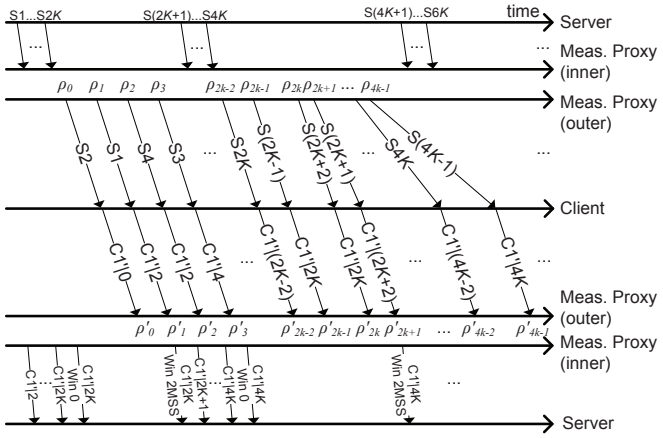


Figure 3: Two probe rounds of sending rates.

ing the accurate available bandwidth in a short period [35]. However, the time required is still too long for seeking a suitable video quality level. In fact, it is not necessary to probe the network with all possible sending rates for deciding whether a video quality level can be smoothly played by the client under the current network condition. Instead, we propose probing the network with a set of selected sending rates that correspond with the bit rate of quality levels. By doing so, we can quickly know whether a sending rate is supported by the network through the obtained RTT samples.

We adjust the packet sending rate by varying the inter-departure time (IDT) of packets in each probing round. The IDT γ_k of packet sending rate λ_k is computed by

$$\gamma_k = (W \times 8)/\lambda_k, k = 0, 1, ... \quad (1)$$

The RTT samples can be computed as the duration from sending a TCP data packet to receiving a corresponding acknowledgement packet (i.e., ACK). However, the TCP delayed acknowledgement mechanism [14] may bias the results because it allows the receiving side to acknowledge multiple data packets with a single TCP ACK. To address this problem, the measurement proxy reorders the data packets by pairs, so that clients send an ACK packet for every TCP data packet. After the reordering, the packet sending sequence for the first probing round is $\{S2,\ S1,\ S4,\ S3, ..., S2K, S(2K-1)\}$ with the sending time $\{\rho_0, \rho_1, ..., \rho_{2k-1}\}$. Assume there is no packet loss, the response packets for the first probing round are $\{C1'|0,\ C1'|2,\ C1'|2,\ C1'|4, ..., C1'|(2K-2), C1'|2K\}$ and arrive the measurement proxy at time $\{\rho'_0, \rho'_1, ..., \rho'_{2k-1}\}$, respectively. Hence, the RTT estimates are retrieved by $\{\rho'_0 - \rho_0, \rho'_1 - \rho_1, ...\}$.

4.3 Determining the Video Quality Levels

Before conducting the measurement, we can first measure the network capacity of the path using existing tools, like TRIO [15]. After that, we use the sending rate that is equal to the capacity to do a round of measurement and calculate the loss rate (i.e., U_{max}), the average RTT (i.e., T_{max}), and the variance of RTT (i.e., V_{max}). Since the available bandwidth should be less than or equal to the capacity, the values $\{U_{max}, T_{max}$, and $V_{max}\}$ serve as the upper bound of these metrics. Similarly, we use the minimal sending rate to perform another round of measurement and compute the loss rate (i.e., U_{min}), mean RTT (i.e., T_{min}), and variance (i.e., V_{min}). Since we assume that the available bandwidth is larger than the minimal sending rate, these values $\{U_{min}, T_{min}$, and $V_{min}\}$ are the lower bound of the metrics.

When the sending rate is higher than the available bandwidth, the probability of packet loss increases [22]. An acceptable sending rate leads to a smaller packet loss rate than U_{max}. Using a packet sending rate higher than the available bandwidth will inflate the mean RTT and the RTT variance, both of which severely affect the average throughput of network path. In contrast, a lower RTT and RTT variance can be observed when a lower packet sending rate is used. The performance will be acceptable to users. Therefore, we adopt a conservative approach that bounds the mean RTT and the RTT variance to determine whether a sending rate is acceptable. More precisely, besides the loss rate should be smaller than U_{max}, an acceptable sending rate should result in a mean RTT less than $T_{min} + a \times V_{min}$ and a variance less than $b \times V_{max} + c \times V_{min}$. In our experiments, we set $a = 3$ and $b = c = 1/2$. The highest acceptable packet sending rate will be the *preferred rate*.

After each round of measurement, the measurement proxy informs the latest preferred rate to clients for quality adaptation. If clients selects a quality level with bit rate lower than the preferred rate, video rebuffering is unlikely to occur. However, sometimes, we choose quality level with bit rate higher than the preferred rate for a short period of time. We will elaborate our quality adaptation algorithm, QDASH-qoe, in section 5.

4.4 Evaluating QDASH-abw

4.4.1 Experiment Setup

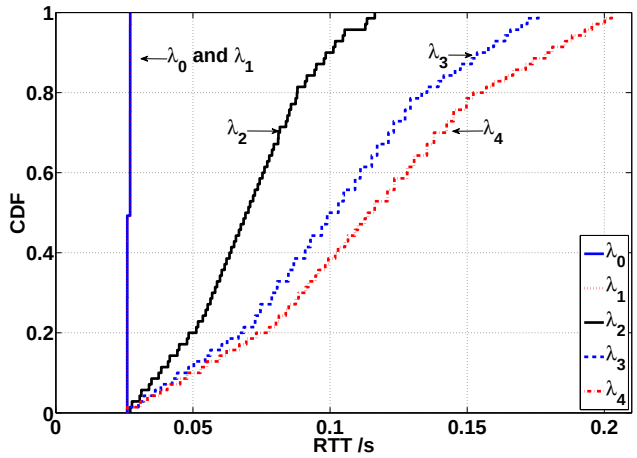
We have implemented a prototype of measurement proxy by using a Linux bridge. The QDASH-abw module hooks to the bridge with `NFQUEUE` target in `iptables` [4]. This design enables the QDASH-abw module to capture, discard, or manipulate all the packets flowing between the video server and clients. We inject a TCP MSS option with value W in the TCP SYN packet such that the server sends W-byte video data packets [27]. Once the QDASH-abw module recognizes an `HTTP GET`, it buffers and resends video data packets according to the probing methodology described in section 4.2.

In the following evaluation, we adopted a dumbbell client-server architecture. A click modular router [23] was placed between the video server and the client to control the available bandwidth. The packet size W and the packet train length, K were set to 576 bytes and 25, respectively. The packet sending rates, $\lambda \in \{\lambda_0, \lambda_1, \lambda_2, \lambda_3, \lambda_4\}$, were respectively selected as $\{0.3, 0.7, 1.5, 2.5, 3.5\}$ Mbps which are the bit rate of video quality levels adopted by Akamai Adaptive Video Streaming [18]. The client uses `wget` [8] to send `HTTP GET` requests to download an MP4 video file from the server. For better illustrating the variations of RTT in different packet sending rate during the whole experiments, the sending rates were scheduled in a round-robin manner.

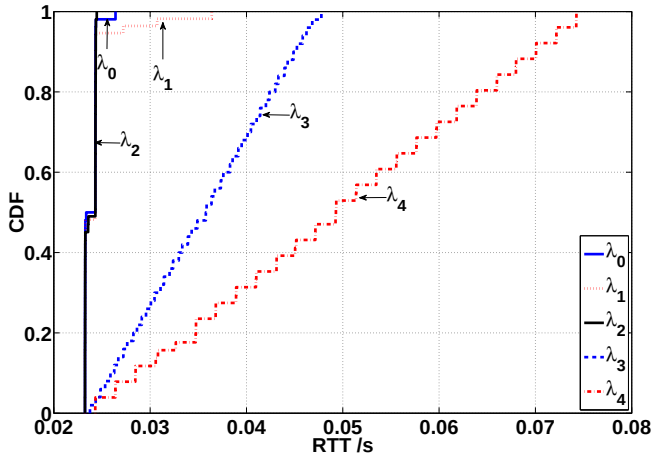
4.4.2 Experiment Result

Figure 4 shows the CDF of sampled RTTs under different available bandwidth. In Figure 4(a), the available bandwidth is 1 Mbps. When λ_0 and λ_1 are used to measure the available bandwidth, the RTTs are similar and do not have large variance compared to the results from other sending rates. The reason is that λ_0 and λ_1 are less than the avail-

able bandwidth and the packets do not suffer from a long queuing delay. On the other hand, when using λ_2 to λ_4, we can observe much longer RTTs with larger variance. The reason is that the bottleneck needs much more time to process the packets and the queueing delay increase. In Figure 4(b), the available bandwidth is 2 Mbps and we can observe similar results as Figure 4(a). The only difference is that since λ_2 is less than 2 Mbps the corresponding RTTs are similar to those resulted from λ_0 and λ_1.



(a) 1 Mbps



(b) 2 Mbps

Figure 4: CDF of sampled RTTs under available bandwidth of 1 Mbps and 2 Mbps.

Figure 5 illustrates the mean and the variance (expressed in standard deviations, std) of sampled RTTs with different packet sending rate, λ, and different available bandwidth. The number on the x-axis, $\{0, ..., 4\}$, represents the packet sending rate, $\{\lambda_0, ..., \lambda_4\}$, respectively. For each available bandwidth, our system conducts the measurement with different packet sending rate. The x-axis indicates the sending rate (λ_0 to λ_4) and the y-axis is the mean RTT. We plot the standard deviations (std) of the RTT along with its mean values.

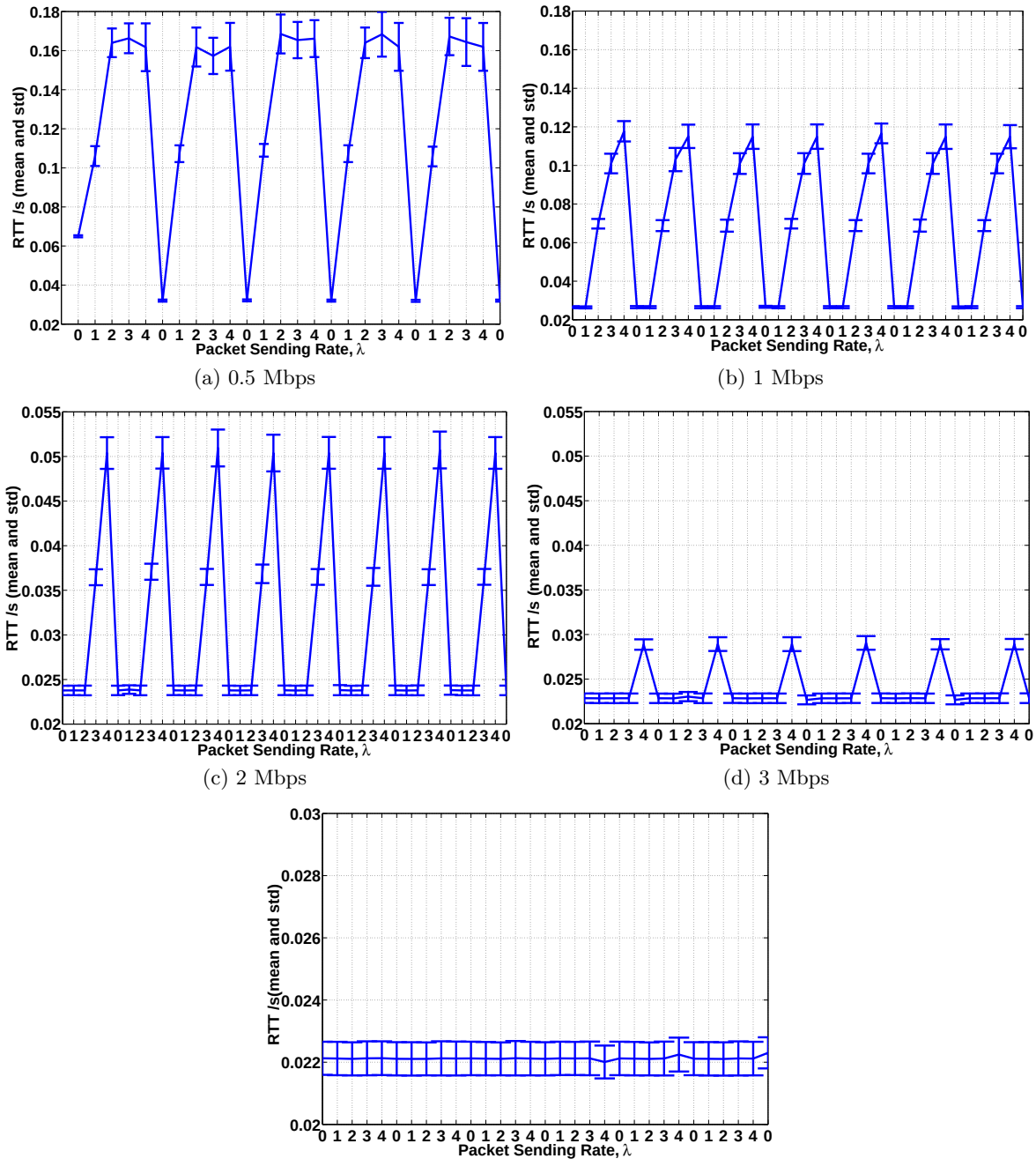
In Figure 5(a), since the available bandwidth is 0.5 Mbps, all except λ_0 resulted in high average RTT with large std. When the available bandwidth increases to 1 Mbps as shown in Figure 5(b), the average RTTs and the corresponding std resulted from λ_2 to λ_4 are still larger than that of λ_0 and λ_1 because λ_2 to λ_4 are larger than the available bandwidth. However, compared to Figure 5(a), the average RTT and the std for λ_2 to λ_4 decrease. In Figure 5(c) and 5(d), the available bandwidth becomes 2 Mbps and 3 Mbps, respectively. From these two figures, we can still observe that the rate(s) higher than the available bandwidth led to larger mean RTT and std than the rates less than the available bandwidth. However, such difference decreases with the increase of the available bandwidth. In Figure 5(e), the available bandwidth is 5 Mbps larger than all packet sending rates. Therefore, the average RTTs and the std from different sampling rate are similar.

We adopted available bandwidth changing profiles in [11] to simulate changes in network conditions and examine the behaviors of QDASH-abw. Figure 6 shows the time series of mean RTT of different packet sending rates in three profiles – persistent variations, short term variations with positive spikes, and short term variations with negative spikes. The black dashed line, with the legend A, in the figures plot the emulated network available bandwidth.

Figure 6(a) plots the results and the emulated available bandwidth for a persistent variations. We emulate this condition by sequentially limit the available bandwidth to 5 Mbps for 20 seconds, 1 Mbps for 20 seconds, 5 Mbps for 30 seconds, and finally 2 Mbps until the experiment ends. For the first 20 seconds, the available bandwidth is sufficient for all the sending rates. The RTTs are close to the emulated delay (20 ms). During the period of 20 to 40 seconds, the sending rates λ_2 to λ_4 are higher than the available bandwidth. The RTTs for these three sending rates inflate at least 3 times than the RTT in the previous period. Then, the RTTs for all the sending rates fall back to around 20 ms after the available bandwidth restored to 5 Mbps. After 70 seconds, the available bandwidth is 2 Mbps, which is higher than λ_0 to λ_2, but lower than that of λ_3 to λ_4. Hence, we only observe inflated RTTs for λ_3 and λ_4.

Figure 6(b) shows a case of short term variations with positive spikes. The available bandwidth for most of the time in the experiment is 1 Mbps, except there is a 2-second spike to 5 Mbps and a 5-second spike to 10 Mbps at time 30 seconds and 62 seconds respectively. The RTTs for λ_2 to λ_4 are significantly higher than that of λ_0 and λ_1 when the available bandwidth is 1 Mbps. In the 2-second spike, the measured RTTs for λ_3 and λ_4 drops sharply to the same level as λ_0 and λ_1. λ_2 does not show a RTT decrease, because there is no sample with λ_2 during that spike. At the second spike to 10 Mbps, the RTTs for all sending rates fall to the same level, which means the available bandwidth is higher than the highest sending rate.

In Figure 6(c), we illustrate the case of short term variations negative spikes. The available bandwidth at most of the time is 5 Mbps, we emulate three spikes with duration of 2 seconds, 5 seconds, and 10 seconds to 1 Mbps at the time 30 seconds, 62 seconds, and 97 seconds, respectively. In the 2-second spike, the rates λ_0 and λ_2 shows RTT inflation. But, the increase in RTT for λ_0 is due to packet loss during the moment of switching the available bandwidth setting in the click router. Other sending rates are not scheduled during this short spike. For the spike longer than 5 seconds, our measurement can capture significant RTT inflation for the sending rate higher than available bandwidth.



(a) 0.5 Mbps

(b) 1 Mbps

(c) 2 Mbps

(d) 3 Mbps

(e) 5 Mbps

Figure 5: The mean and the standard deviation of sampled RTTs with different sampling rate and different available bandwidth.

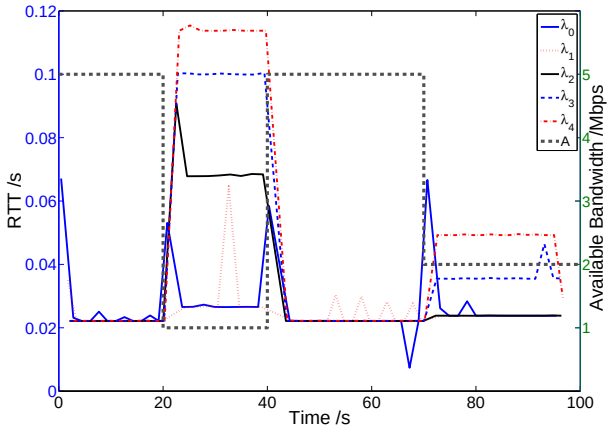
4.5 Summary

QDASH-abw provides a novel probing methodology to determine the highest quality level the network can support. It fastens the existing available bandwidth measurements by sending probes with a few selected packet sending rate. Our evaluation shows that QDASH-abw is accurate and is sensitive to available bandwidth variations. With a timely estimation of the network condition, the video player at the client can better select a suitable video quality levels to download.

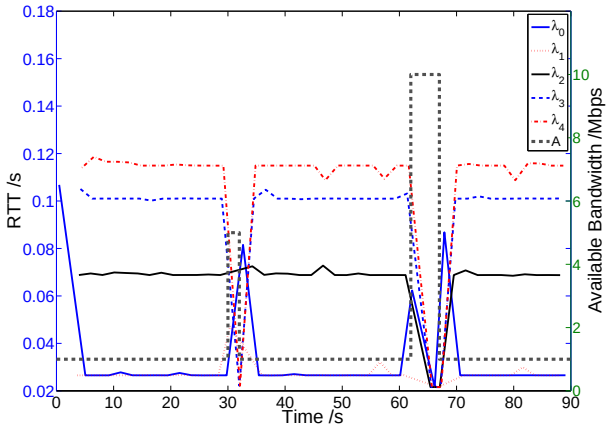
5. A QOE-AWARE SWITCHING ALGORITHM

5.1 Buffer-aware Strategies

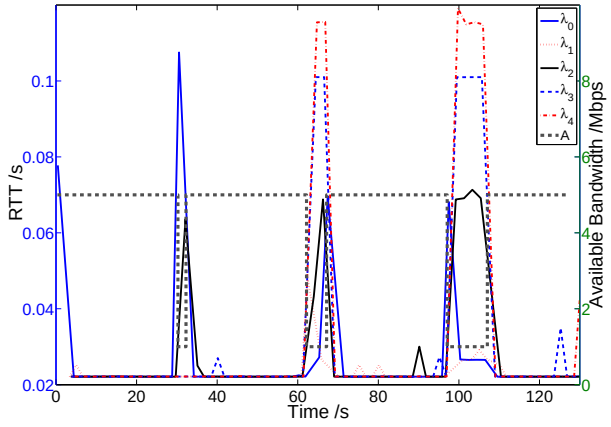
Similar to other streaming technologies, such as UDP streaming and classic HTTP streaming, a buffer is deployed to reduce the chance of playback interruptions. However, it is not feasible to stream videos at a higher bit rate than the end-to-end available bandwidth. In this case, congestion and packet loss will occur at the bottleneck link. As a result, spatial artifacts, such as blocking frames, are often seen due



(a) Persistent variations



(b) Short term variations – positive spikes



(c) Short term variations – negative spikes

Figure 6: Evaluations with three network profiles.

to incomplete video decoding.

DASH transfers video data via TCP/IP. The reliable service provided by TCP ensures the integrity of video data. Video with a higher bit rate than the network throughput can be delivered without loss of picture quality. Even though the download rate is lower than the playback rate, interruption-free playback can be sustained for a short period of time by using the buffer. During this short time period, we can first request the video fragments with quality level between the original and the final level to append to the video buffer. By inserting this intermediate quality levels, the original and final (lower) quality levels can be bridged.

Figure 7 gives an example of how we can utilize the buffer in quality level transitions. The y-axis at the left is the bit rate of the different quality levels, while the y-axis at the right is the measured network throughput. The x-axis represents the time. This example illustrates a video encoded into a set of five quality levels, $L \in \{l_0, l_1, ..., l_4\}$. The solid line, dashed line, and the dot-dashed line represent the video levels requested by the video player, the network throughput, and the quality level shown to the user, respectively.

At time t_0, we assume the video buffer of size B is full. The video player plays and requests video fragments at quality level l_4. However, the network throughput decreases to a level that is barely higher than the bit rate of quality level l_1 at t_1. After the video player detects the change at t_2, existing adaptation algorithms [11] switch the quality level to l_1 in order to adapt to the network conditions. As a result, users would perceive a sudden decline in picture quality. In order to avoid this, we propose to insert an intermediate level, for example l_2, between l_4 and l_1. With the same scenario, the download rate is higher than the playback rate (i.e. the time to download the fragments,$(t_3 - t_2)$, is longer than the length of fragments $(t_5 - t_4)$). If this situation were maintained for a longer time period, rebuffering would occur. However, while the video player is downloading fragments of l_2, it continues playing the buffered video at the original quality. Therefore, we can use the buffered video playback period, from t_2 to t_4, to download the video fragments at the intermediate level.

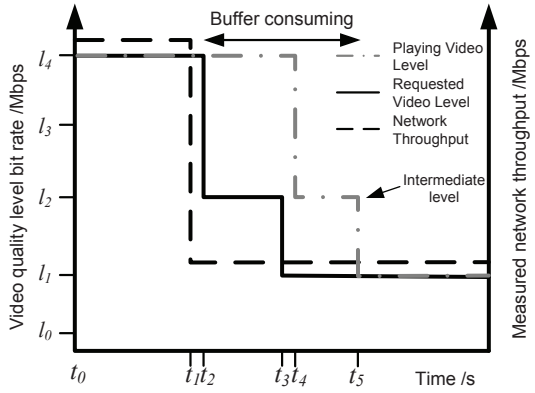


Figure 7: A quality transition.

Although intermediate levels can smooth out the picture quality change, rebuffering events also reduce the QoE [29]. Downloading intermediate quality levels should also avoid causing the buffer starvation. The maximum number of intermediate quality video fragments to be downloaded, n_{frag}, is given by (2).

$$n_{frag} = \lfloor t_{buffer} \frac{\beta}{s_{frag}} \rfloor, \quad (2)$$

$$t_{buffer} = \frac{B}{\left(1 - \frac{\beta}{s_{frag}}\right)}, \quad (3)$$

$$s_{frag} = \lambda_i \times \theta, \quad (4)$$

where λ_i is the average bit rate of the intermediate quality level, l_i with a fragment length of θ seconds of video. B is the video buffer size in second of video, and β is the degraded network throughput. s_{frag} computes the average size of each video fragment of quality level, l_i. t_{buffer} is the time period for which the video buffer is able to offset the download of intermediate quality levels. Therefore, n_{frag} gives the number of complete fragments that can be downloaded within t_{buffer}.

5.2 Evaluating the QoE Impact on Quality Adaptation

In section 5.1, we have discussed how to utilize the buffer to provide an intermediate quality level. Under this model, we can have many possibilities by choosing different intermediate levels, the number of intermediate levels, and buffer sizes. However, the perception to users for each case is not the same. We assess the QoE on the cases to identify the impact of each factor.

To assess the QoE, common approaches employ objective metrics, such as PSNR (Peak signal-to-noise ratio) and carrying out subjective experiments. However, objective metrics are not useful in our case, because objective metrics only consider the spatial quality, but not the transition of quality. Therefore, we perform subjective assessments of human subjects to evaluate the overall perceived quality (i.e., the QoE).

As quality degradations are usually noticeable and cause larger impact to the QoE [17], we carry out subjective assessments on their perception of quality adaptation which is possible under the same network conditions. To ensure the consistency among subjects, we pre-define a set of rule sets to profile the quality level changes. Then, the subjects rate the QoE on different rule sets in terms of MOS.

5.2.1 Experiment Setup

We have implemented an experimental video delivery platform to carry out subjective assessments. The video server is installed with Debian 6.0, Apache web server 2.2.16, and Adobe's HTTP Origin Module [9] in order to support Adobe Dynamic Streaming. In addition, the video player is developed using the OSMF (Open Source Media Framework) [5] and SMP (Strobe Media Playback) [6]. We replace the original quality adaption algorithm in the OSMF, so that the quality level switching follows a pre-defined scenario.

We defined each scenario with a rule set, $\mathbb{R}$, expressed by (5). Each rule set contains at least one rule tuple represented by $< \ell, d >$, where ℓ is the quality level and d is the number of bytes to be downloaded at that quality level. One rule tuple is completed when the number of bytes downloaded at the quality level is greater than or equal to the defined value. The player proceeds to the next rule tuple until all the rule tuples have been used. At this point, the quality level stays at the last level. Therefore, we can emulate different kinds of quality changes by defining these rule sets.

$$\mathbb{R} = \{< \ell_0, d_0 >, < \ell_1, d_1 >, ..., < \ell_k, d_k >\} \quad , k = 0, 1, 2, ... \tag{5}$$

5.2.2 Generation of Rule Sets

In our experiments, we emulate a sudden drop of throughput from 4 Mbps to 400 Kbps after playing the first three fragments. Therefore, the highest possible quality drop is from l_4 to l_0. We also emulate three buffer sizes, 1 fragment, 3 fragments, and 8 fragments. These three buffer sizes are typical values employed by DASH video players. Then, we compute the maximum number of fragments supported at the intermediate quality levels, n_{frag}, (2). Table 1 lists the values of n_{frag} for our experiments. For 1-fragment buffer, only quality levels below l_2 allow downloading at least one complete fragment during the buffer compensated period.

Three different initial quality levels, l_4, l_2, and l_0, are employed in the experiments. As the subjects have not watched the video before, varying the initial quality levels can give an insight as to how the subjects perceive the picture quality in a no reference context.

Table 2 lists the rule sets we employed in our experiments. Except for $\mathbb{R}_0$, the subjects watch the first three fragments of the video at the original quality level defined in the first tuple of the rule set. Then, the video player plays all the fragments in the buffer to emulate the consumption of the buffer due to a decrease in throughput. After that one or two intermediate quality levels are inserted and played. Finally, the player reaches the target level, l_0.

$\mathbb{R}_0$ is the base case providing the worst video quality. The rule sets denoted with $\mathbb{R}_{1,x}$, $\mathbb{R}_{3,x}$, and $\mathbb{R}_{8,x}$ are of buffer size 1, 3, and 8 fragments, respectively. $\mathbb{R}_{3,8}$, $\mathbb{R}_{3,9}$, and $\mathbb{R}_{3,10}$ have two intermediate levels. Half of the buffer is used to load a higher quality level, and the remaining half is used to load a lower quality level. Figure 8 depicts the quality level transition of $\mathbb{R}_{1,4}$, $\mathbb{R}_{3,7}$, and $\mathbb{R}_{3,8}$. $\mathbb{R}_{1,4}$ and $\mathbb{R}_{3,8}$ have one and two intermediate levels, respectively, while $\mathbb{R}_{3,7}$ starts with quality level l_2. The buffer of $\mathbb{R}_{8,4}$ and $\mathbb{R}_{8,5}$ is large enough to allow the intermediate level to be used until the end of video clip. Therefore, these two rule sets do not switch to quality level l_0.

Table 1: The maximum number of fragments supported at intermediate levels, n_{frag}.

Quality levels	Buffer size 1 (fragment)	3 (fragments)	8 (fragments)
l_4	0	1	4
l_3	0	2	6
l_2	1	4	11
l_1	5	15	42

5.2.3 Video Materials

We prepared 11 short video clips, each with a duration of about 90 seconds. The video clips were taken from various kinds of sources, including sports, movie trailers, animations, and music videos. The quality of the source video was at least equal to that of l_4 in Table 3. We subsequently downsampled the source video clips to other quality levels. To emulate more realistic environment, we used the video quality specification adopted by Akamai Adaptive Video Streaming [1]. Table 3 shows the profiles of all the quality levels [18]. Adobe's File Packager [9] was then used to package the video files of different quality levels and translate the encoded video files into fragments. We define fragment length, θ, be four seconds of video, which is the default value in the File Packager. Therefore, each video clip contains about 23 ($\lceil 90/4 \rceil$) fragments.

Table 2: Experiment rule sets.

$\mathbb{R}_0$	$< l_0, 23 \times s_0 >$
$\mathbb{R}_{1,1}$	$< l_4, 3 \times s_4 >, < l_4, 1 \times s_4 >, < l_2, 1 \times s_2 >, < l_0, 17 \times s_0 >$
$\mathbb{R}_{1,2}$	$< l_4, 3 \times s_4 >, < l_4, 1 \times s_4 >, < l_1, 5 \times s_1 >, < l_0, 13 \times s_0 >$
$\mathbb{R}_{1,3}$	$< l_4, 3 \times s_4 >, < l_4, 1 \times s_4 >, < l_0, 19 \times s_0 >$
$\mathbb{R}_{1,4}$	$< l_2, 3 \times s_2 >, < l_2, 1 \times s_2 >, < l_1, 5 \times s_1 >, < l_0, 13 \times s_0 >$
$\mathbb{R}_{1,5}$	$< l_2, 3 \times s_2 >, < l_2, 1 \times s_2 >, < l_0, 19 \times s_0 >$
$\mathbb{R}_{3,1}$	$< l_4, 3 \times s_4 >, < l_4, 3 \times s_4 >, < l_4, 1 \times s_4 >, < l_0, 16 \times s_0 >$
$\mathbb{R}_{3,2}$	$< l_4, 3 \times s_4 >, < l_4, 3 \times s_4 >, < l_3, 2 \times s_3 >, < l_0, 15 \times s_0 >$
$\mathbb{R}_{3,3}$	$< l_4, 3 \times s_4 >, < l_4, 3 \times s_4 >, < l_2, 4 \times s_2 >, < l_0, 13 \times s_0 >$
$\mathbb{R}_{3,4}$	$< l_4, 3 \times s_4 >, < l_4, 3 \times s_4 >, < l_1, 15 \times s_1 >, < l_0, 2 \times s_0 >$
$\mathbb{R}_{3,5}$	$< l_4, 3 \times s_4 >, < l_4, 3 \times s_4 >, < l_0, 17 \times s_0 >$
$\mathbb{R}_{3,6}$	$< l_2, 3 \times s_2 >, < l_2, 3 \times s_2 >, < l_1, 15 \times s_1 >, < l_0, 2 \times s_0 >$
$\mathbb{R}_{3,7}$	$< l_2, 3 \times s_4 >, < l_2, 3 \times s_4 >, < l_0, 17 \times s_0 >$
$\mathbb{R}_{3,8}$	$< l_4, 3 \times s_4 >, < l_4, 3 \times s_4 >, < l_3, 1 \times s_3 >, < l_2, 2 \times s_2 >, < l_0, 14 \times s_0 >$
$\mathbb{R}_{3,9}$	$< l_4, 3 \times s_4 >, < l_4, 3 \times s_4 >, < l_3, 1 \times s_3 >, < l_1, 7 \times s_1 >, < l_0, 9 \times s_0 >$
$\mathbb{R}_{3,10}$	$< l_4, 3 \times s_4 >, < l_4, 3 \times s_4 >, < l_2, 1 \times s_2 >, < l_1, 7 \times s_1 >, < l_0, 9 \times s_0 >$
$\mathbb{R}_{8,1}$	$< l_4, 3 \times s_4 >, < l_4, 7 \times s_4 >, < l_4, 4 \times s_4 >, < l_0, 9 \times s_0 >$
$\mathbb{R}_{8,2}$	$< l_4, 3 \times s_4 >, < l_4, 7 \times s_4 >, < l_3, 6 \times s_3 >, < l_0, 7 \times s_0 >$
$\mathbb{R}_{8,3}$	$< l_4, 3 \times s_4 >, < l_4, 7 \times s_4 >, < l_2, 11 \times s_2 >, < l_0, 2 \times s_0 >$
$\mathbb{R}_{8,4}$	$< l_4, 3 \times s_4 >, < l_4, 7 \times s_4 >, < l_1, 13 \times s_1 >$
$\mathbb{R}_{8,5}$	$< l_2, 3 \times s_2 >, < l_2, 7 \times s_2 >, < l_1, 13 \times s_1 >$

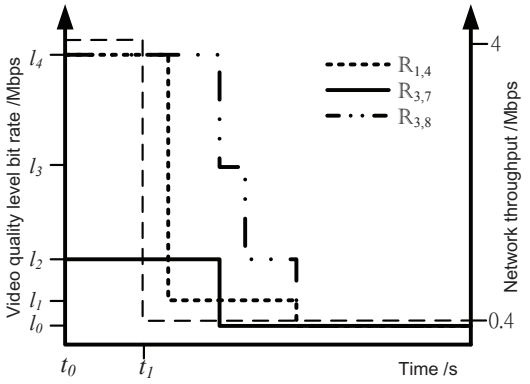


Figure 8: The quality transition of $\mathbb{R}_{1,4}$, $\mathbb{R}_{3,7}$, and $\mathbb{R}_{3,8}$.

Table 3: Profiles of the video quality levels [18].

Parameters	l_0	l_1	l_2	l_3	l_4
Video height (pixel)	180	360	360	720	720
Video width (pixel)	320	640	640	1280	1280
Avg. video bit rate (kbps)	300	700	1500	2500	3500
Avg. audio bit rate (kbps)	160	128	128	128	128
Video frame rate (fps)	29.97	29.97	29.97	29.97	29.97
Video codec	H.264	H.264	H.264	H.264	H.264
Audio codec	AAC	AAC	AAC	AAC	AAC

5.2.4 Subject Assessment

Subjects were told that a sudden drop in network throughput was emulated during each experiment. They were advised to watch the video in full screen, and not to pause or time-shift the video playback. Each subject was asked to watch 11 video clips, as mentioned in Section 5.2.3. We applied one of the experiment rule sets from Table 2 to the video playback. $\mathbb{R}_0$ was shown to all subjects. For the other ten experiment sessions, the video player randomly selected one of the experiment rule sets. Therefore, the sequence for all subjects was randomized to mitigate the order effect.

After completing each playback, subjects were immediately required to answer questions on the video playback they just watched. We first asked the subject if they notice any quality change in video quality during the playback. Then, the subjects were asked to separately rate the perceived quality on the following aspects – picture quality, sound quality, playback smoothness, and video content. Finally, they were asked to give a composite score on the overall perceived quality. We adopt a 7-point Likert scale, from '1' (Bad) to '7' (Excellent), to measure the MOS for higher granularity.

5.3 Assessment Results

5.3.1 Descriptive Statistic

A total of 24 subjects, 19 males and 5 females, participated in the subject assessment. All the subjects were volunteers, and non-experts in video quality assessment, with normal vision. They have the basic computer skills to use the experiment platform. We obtained 242 valid samples of rating on overall perceived quality. Figure 9 depicts the frequency distribution of the overall QoE rating. No subject rated '1' and only two experiment sessions gave a rate of '7'. We think this is reasonable as there was no service interruption and only quality degradation took place in all cases. We have also validated that all the rule sets and the choice of video were evenly distributed among all the samples.

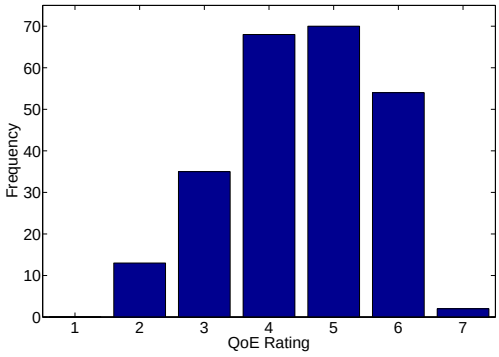


Figure 9: The overall distribution of overall QoE rating.

5.3.2 Comparisons on MOSes

We compare the QoE between all the rule sets by computing the MOS difference as given by (6).

$$\Delta MOS(i,j) = MOS(\mathbb{R}_i) - MOS(\mathbb{R}_j) \quad (6)$$

,where $MOS(\mathbb{R}_j)$ is the mean QoE rating obtained using rule set $\mathbb{R}_j$. In addition, we also determine the significance level of the MOS difference, p, by using Independent-samples t-test. Table 4 shows the MOS difference between rule sets, ΔMOS(column,row). Due to space constraints, we only show the MOS differences at the significant level less than 0.1 (p <0.1).

Our results indicate that, $\mathbb{R}_{3,4}$, $\mathbb{R}_{8,4}$, and $\mathbb{R}_{8,5}$ obtain a negative MOS difference with respect to most of the other rule sets in the respective rows, meaning that these three rule sets have a higher MOS compared to other cases and outer-perform the other rule sets in terms of QoE. However, no significant difference in MOS is observed among these three cases. $\mathbb{R}_{8,4}$ and $\mathbb{R}_{8,5}$ do not drop to the lowest quality level, l_0, at the end of the video playback. Therefore, the higher QoE could be due to the higher final picture quality. The MOSes of $\mathbb{R}_{8,4}$, and $\mathbb{R}_{8,5}$ are significantly higher than that of $\mathbb{R}_{8,1}$ which plays the longest time in highest quality level. This shows that providing as high video quality as possible does not lead to the highest QoE. Surprisingly, $\mathbb{R}_0$, which plays at the lowest quality level throughput the whole playback, does not obtain the lowest MOS. As there is no quality transitions, the subjects did not know they were watching the video with the lowest quality.

Rule sets without intermediate quality levels usually have lower perceived quality. One example is that $\mathbb{R}_{3,5}$ has lower MOS than six other rule sets with intermediate levels. However, using two intermediate levels ($\mathbb{R}_{3,8}$, $\mathbb{R}_{3,9}$, $\mathbb{R}_{3,10}$) shows only slight impact in improving the QoE. Among the three rule sets, the QoE for $\mathbb{R}_{3,8}$ is significantly higher than five other rule sets.

On the other hand, large video buffer size is not necessary improve the QoE. Many rule sets with small buffer size have no significant difference in MOS to those with large buffer size. One example is $\mathbb{R}_{3,4}$ which provides better perceived quality than $\mathbb{R}_{8,1}$. However, larger video buffer consumes more resources at the client-side, such as cache memory.

To summarize the results, inserting intermediate levels usually gives a better QoE, while one intermediate level gives the highest perceived quality. A small video buffer can only have short period to have intermediate level, and usually produces lower QoE.

5.4 Designing an QoE-aware Switching Algorithm

From the results obtained in section 5.3.2, the QoE can be increased by inserting intermediate levels in between quality level down switching. Hence, we formulate and propose a QoE-aware switching algorithm, which is shown in Algorithm 2, by considering the intermediate levels and video buffer size.

This algorithm is run before deciding the quality level of the next video fragment. We obtain the supported quality level, $l_{support}$, by using QDASH-abw. If $l_{support}$ is lower than the current quality level, l_{cur}, by two levels, we compute the number of fragments of intermediate levels to be downloaded, n_{frag}, by using (2)-(4). We choose the intermediate level as one level above the target quality level. So, the period of watching in intermediate level can be maximized. It also shows effective in the QoE assessment in section 5.3.2.

Algorithm 2 A QoE-aware quality adaptation algorithm

$l_{support}$: The quality level current network condition can support
l_{cur}: Current quality level
l_{nxt}: Proposed quality level
t_{buffer}: Number of intermediate quality video fragments to be downloaded
B: Buffer size in video second
$b(l)$: Bit rate of quality level l
$s(l,\theta)$: Average size of 1 video fragment at quality level l
n_{frag}: Number of fragments for the proposed quality level
if $l_{support} < l_{cur}$ **then**
  if $(l_{cur} - l_{support}) > 1$ **then**
    $t_{buffer} \leftarrow \frac{B}{1-b(l_{support})/s(l_{support},\theta)}$
    $n_{frag} \leftarrow \lfloor t_{buffer} \times b(l_{support})/s(l_{support},\theta) \rfloor$
    if $n_{frag} > 0$ **then**
      $l_{nxt} \leftarrow l_{support} + 1$
    else
      $l_{nxt} \leftarrow l_{support}$
    end if
  else
    $l_{nxt} \leftarrow l_{support}$
  end if
else
  $l_{nxt} \leftarrow l_{support}$
end if

6. RELATED WORKS

Application level throughput measurement is widely adopted for adapting the video quality. However, throughput measurement can be largely fluctuated by packet losses. Liu et al. [26] proposed an adaption algorithm based on smoothed network throughput to mitigate these fluctuations. However, the smoothing technique reduce the sensitivity of results.

Besides, adaptation requiring modification of servers are usually harder to deploy to existing DASH systems. Kuschnig et al. [24] evaluated three adaptation schemes for DASH of H.264/SVC – application-layer bandwidth estimation, TCP stack-based bandwidth estimation, and deadline-driven adaptive streaming. The latter two algorithms are harder to deploy and are not compatible with existing DASH system. They require to re-encode the existing H.264/AVC videos and modify the video server. Feedback control is introduced in [19] to feedback information to the video server for adjusting the quality levels and its sending buffer at the server-side.

Cranley et al. [16, 17] investigated the user perception on quality adaptation and proposed the Optimal Adaptation Trajectory (OAT) to maximize the QoE of UDP-based video streaming. UDP-based video streaming degrades the picture quality of video in poor network conditions, while DASH, which is TCP-based, reflects in rebuffering. The user perception can be very different between both types of streaming.

7. CONCLUSION

In this paper, we proposed a comprehensive QoE-aware DASH system - QDASH. It consists of two modules. QDASH-abw is a novel probing methodology which is tailor-made for DASH system to measure the network. By reducing the choice of packet sending rate, QDASH-abw is sensitive to

Table 4: The MOS difference, ΔMOS(column,row).

	$\mathbb{R}_0$	$\mathbb{R}_{1,1}$	$\mathbb{R}_{1,2}$	$\mathbb{R}_{1,3}$	$\mathbb{R}_{1,4}$	$\mathbb{R}_{1,5}$	$\mathbb{R}_{3,2}$	$\mathbb{R}_{3,3}$	$\mathbb{R}_{3,5}$	$\mathbb{R}_{3,6}$	$\mathbb{R}_{3,7}$	$\mathbb{R}_{3,10}$	$\mathbb{R}_{8,1}$	$\mathbb{R}_{8,2}$
$\mathbb{R}_{3,1}$							-1.03†							
$\mathbb{R}_{3,3}$							-0.70†							
$\mathbb{R}_{3,4}$	-1.03*	-1.25*	-1*	-0.88†	-1.23**	-0.93*	-1.67**	-0.97*	-1.25**	-1.03*		-1.21*	-0.98*	
$\mathbb{R}_{3,7}$		-0.92†			-0.9†		-1.33*		-0.92†					
$\mathbb{R}_{3,8}$		-0.92†			-0.9*		-1.33**		-0.92*			1.12†		
$\mathbb{R}_{3,9}$							-1.23†		-0.82†					
$\mathbb{R}_{3,10}$											0.88†			
$\mathbb{R}_{8,2}$							-1.03*							
$\mathbb{R}_{8,3}$								-1†						
$\mathbb{R}_{8,4}$	-1.12*	-1.33**	-1.10*	-0.96*	-1.32**	-1.02*		-1.75***	-1.05**	-1.33**	-1.12*	-1.29**	-1.06*	-0.72†
$\mathbb{R}_{8,5}$		-1.52*	-1.27*	-1.15†	-1.5*	-1.2*		-1.93*	-1.23*	-1.52**	-1.3*	-1.48*	-1.24†	

Note: $\dagger p < 0.1$, $*p < 0.05$, $**p < 0.01$, $***p < 0.001$

the change in available bandwidth and provides accurate decisions on which the video quality levels are supported by current network conditions. On the other hand, we examined the effects of QoE by video quality adaptation. We found that users is favored to have intermediate levels in between quality drops. We thus proposed QDAHS-qoe, a QoE-aware quality adaptation algorithm.

8. ACKNOWLEDGMENTS

We thank the three anonymous reviewers for their critical comments, Michael Cumming for editing the paper, and all the participants in the experiments. This work is partially supported by a grant (ref. no. ITS/355/09) from the Innovation Technology Fund in Hong Kong and a grant (ref. no. H-ZL17) from the Joint Universities Computer Centre of Hong Kong.

9. REFERENCES

[1] Akamai HD Video Demo. `http://wwwns.akamai.com/hdnetwork/demo/flash/default.html`.
[2] comScore releases May 2010 U.S. online video rankings. `http://www.comscore.com/Press_Events/Press_Releases/2010/6/comScore_Releases_May_2010_U.S._Online_Video_Rankings`.
[3] F4V/FLV Technology Center. `http://www.adobe.com/devnet/f4v.html`.
[4] Netfilter. `http://www.netfilter.org`.
[5] Open Source Media Framework (OSMF). `http://www.osmf.org`.
[6] Strobe Media Playback. `http://sourceforge.net/adobe/smp`.
[7] Tcpdump/libpcap. `http://www.tcpdump.org`.
[8] wget. `http://www.gnu.org/s/wget`.
[9] Adobe. HTTP Dynamic Streaming on the Adobe Flash Platform. `http://www.adobe.com/products/httpdynamicstreaming`.
[10] Adobe. Video technology center, delivery: Progressive download. `http://www.adobe.com/devnet/video/progressive.html`.
[11] S. Akhshabi, A. Begen, and C. Dovrolis. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP. In *Proc. ACM MMSys*, 2011.
[12] Apple. HTTP Live Streaming. `http://developer.apple.com/resources/http-streaming`.
[13] A. Begen, T. Akgul, and M. Baugher. Watching video over the Web: Part 1: Streaming Protocols. *IEEE Internet Comput.*, 15(2):54 – 63, 2011.
[14] R. Braden. RFC1122. Requirements for Internet Hosts – Communication Layers, 1989.
[15] E. Chan, A. Chen, X. Luo, R. Mok, W. Li, and R. Chang. TRIO: Measuring asymmetric capacity with three minimum round-trip times. In *Proc. ACM CoNext*, 2011.
[16] N. Cranley, L. Murphy, and P. Perry. User-perceived quality-aware adaptive delivery of MPEG-4 content. In *Proc. ACM NOSSDAV*, 2003.
[17] N. Cranley, P. Perry, and L. Murphy. User perception of adapting video quality. *Int. Journal of human-computer studies*, 64(8):637 – 647, 2006.
[18] L. De Cicco and S. Mascolo. An experimental investigation of the Akamai adaptive video streaming. In *Proc. USAB*, 2010.
[19] L. De Cicco, S. Mascolo, and V. Palmisano. Feedback control for adaptive live video streaming. In *Proc. ACM MMSys*, 2011.
[20] N. Feamster, D. Bansal, and H. Balakrishnan. On the interactions between layered quality adaptation and congestion control for streaming video. In *Proc. Packet Video*, 2001.
[21] M. Jain and C. Dovrolis. End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput. *IEEE/ACM Trans. on Networking*, 11(4):537 – 549, 2003.
[22] L. Kleinrock. *Queueing Systems. Volume 1: Theory.* Wiley-Interscience, 1975.
[23] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Trans. Comput. Syst.*, 18(3):263–297, 2000.
[24] R. Kuschnig, I. Kofler, and H. Hellwagner. An evaluation of TCP-based rate-control algorithms for adaptive Internet streaming of H.264/SVC. In *Proc. ACM MMSys*, 2010.
[25] A. Lie and J. Klaue. Evalvid-RA: trace driven simulation of rate adaptive MPEG-4 VBR video. *ACM Multimedia Systems Journal*, 14:33–50, 2008.
[26] C. Liu, I. Bouazizi, and M. Gabbouj. Rate adaptation for adaptive HTTP streaming. In *Proc. ACM MMSys*, 2011.
[27] X. Luo, E. Chan, and R. Chang. Design and implementation of TCP data probes for reliable and metric-rich network path monitoring. In *Proc. USENIX Annual Tech. Conf. 2009*, 2009.
[28] Microsoft. IIS Smooth Streaming technical overview. `http://learn.iis.net/page.aspx/626/smooth-streaming-technical-overview/`.
[29] R. Mok, E. Chan, and R. Chang. Measuring the quality of experience of HTTP video streaming. In *Proc. IEEE/IFIP IM (pre-conf.)*, 2011.
[30] R. Mok, E. Chan, X. Luo, and R. Chang. Inferring the QoE of HTTP video streaming from user-viewing activities. In *Proc. of ACM SIGCOMM W-MUST*, 2011.
[31] M. Pinson and S. Wolf. Comparing subjective video quality testing methodologies. *Visual Communications and Image Processing*, 5150(1):573 – 582, 2003.
[32] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen. Bitrate and video quality planning for mobile streaming scenarios using a GPS-based bandwidth lookup service. In *IEEE ICME*, 2011.
[33] H. Schwarz, D. Marpe, and T. Wiegand. Overview of the scalable video coding extension of the H.264/AVC standard. *IEEE Trans. on Circuits and Systems for Video Technology*, 17(9):1103 – 1120, 2007.
[34] T. Stockhammer. Dynamic adaptive streaming over HTTP –: Standards and design principles. In *Proc. ACM MMSys*, 2011.
[35] X. Xing, J. Dang, S. Mishra, and X. Liu. A highly scalable bandwidth estimation of commercial hotspot access points. In *Proc. IEEE INFOCOM*, 2011.
[36] M. Zink, O. Künzel, J. Schmitt, and R. Steinmetz. Subjective impression of variations in layer encoded videos. In *Proc. IWQoS*, 2003.