A taxonomy of the parameters used by decision methods for adaptive video transmission

Eugen Dedu, Wassim Ramadan, Julien Bourgeois

Multimedia Tools and Applications manuscript No.
(will be inserted by the editor)

A taxonomy of the parameters used by decision methods for adaptive video transmission

Eugen Dedu · Wassim Ramadan · Julien Bourgeois

Received:

Abstract Nowadays, video data transfers account for much of the Internet traffic and a huge number of users use this service on a daily base. Even if videos are usually stored in several bitrates on servers, the video sending rate does not take into account network conditions which are changing dynamically during transmission. Therefore, the best bitrate is not used which causes sub-optimal video quality when the video bitrate is under the available bandwidth or packet loss when it is over it. One solution is to deploy adaptive video, which adapts video parameters such as bitrate or frame resolution to network conditions. Many ideas are proposed in the literature, yet no paper provides a global view on adaptation methods in order to classify them. This article fills this gap by discussing several adaptation methods through a taxonomy of the parameters used for adaptation. We show that, in the research community, the sender generally takes the decision of adaptation whereas in the solutions supported by major current companies the receiver takes this decision. We notably suggest, without evaluation, a valuable and realistic adaptation method, gathering the advantages of the presented methods.

Keywords Video content adaptation, rate control, congestion control, video streaming

1 Introduction

In recent years, the number of videos pre-encoded in several bitrates has significantly increased to become accessible to everybody such that video service providers like YouTube and DailyMotion became very popular. Videoconference usage is also increasing, and the quality of video used during videoconference sessions can vary

W. Ramadan received a grant of PhD thesis from the Ministry of High Education of Syria.

E. Dedu, W. Ramadan and J. Bourgeois
FEMTO-ST institute, DISC department, 4 pl. Tharradin, 25200 Montbéliard, France
Tel: +33 3 81 99 47 75, Fax: +33 3 81 99 47 91
E-mail: {Eugen.Dedu, Wassim.Ramadan, Julien.Bourgeois}@pu-pm.univ-fcomte.fr

greatly depending on the encoding quality chosen by the sender. All these videos are delivered to final users using streaming services. Multimedia streaming services over Internet, as well as the demand for higher quality from final clients are in constant progression and new video standards like HD and 3D [5] are asking for even more bandwidth. On the other hand, smartphones become ubiquitous nowadays but most of the videos currently available do not the match restrictive capabilities of mobile devices.

IP-based networks like Internet (IP = Internet Protocol [37]) remain best-effort networks which do not provide service quality to users. Characteristics such as congestion losses, variation of available bandwidth and jitter are bound deep inside the design of Internet. The available bandwidth changes frequently, driven by two causes: either the network bandwidth itself changes, or the available bandwidth changes.

The first case appears in wireless networks because the radio link quality changes both the available bandwidth and the number of packet losses. Data rate decrease is due to multiple reasons:

- interferences due to environment or to presence of another equipment working on the same range of frequencies, which decrease the signal to noise ratio for a short period;
- mobility (the user goes further or nearer the access point) which, depending on the distance between the mobile and the access point, leads to signal attenuation and might also cause dynamic rate scaling.

This case appears also when an ISP (Internet Service Provider) changes dynamically the bandwidth allocated to a user. In fact, with the increasing number of streaming services on Internet, more and more traffic is generated, which leads ISPs to limit user bandwidth even during a transfer.

The second case is when a variable number of flows share the same path. The bandwidth available for video streaming is also variable for each flow.

So, we have on the one hand videos with different qualities available in the network, and on the other hand a variable network bandwidth to stream them.

Classical video transmission uses a fixed bitrate from the beginning to the end of the transmission. The bandwidth changing poses problems to the video transmission because when bitrate is smaller than bandwidth, the network is underutilised, and when bitrate is higher than bandwidth, packets will be lost. As such, watching videos is not always a comfortable experience, especially when the bandwidth becomes smaller than the bitrate.

As any other transfer on Internet, video transfers use a transport protocol. Using a transport protocol without congestion control, such as UDP (User Datagram Protocol [36]), leads to major drawbacks: flows on the path are not friendly with each other (some flows take much more share than others), many packets can be lost and there are fears that such inelastic transfers can eventually lead to network congestion collapse. Conversely, a transport protocol with congestion control avoids the above problems. However, this also means that the video data sending rate is regulated by the network bandwidth, and not by the application. As the bandwidth is variable, a classical solution is to continuously adapt video sending rate to network conditions, method called *adaptation*.

The adaptation has several advantages over the static bitrate. It helps to take advantage of the whole available bandwidth or to avoid delays/numerous lost packets; this is especially useful for firm real-time transmissions [26], such as videoconference. Also, whereas in the static bitrate case it is difficult, or even impossible, for the user or some program to decide at the beginning of a video transmission which is the best bitrate to be used during the whole transmission, in the adaptation case the choice of the bitrate is automatic, i.e. it does not involve any action from the user.

The adaptation can be done in several ways. Usually, the video bitrate is increased or decreased each time the available bandwidth increases or decreases. Such adaptation, known in the literature as "rate adaptive video control", can be done by controlling video parameters such as quantization parameter, number of frames per second (FPS) and image size [6].

Even if the rate control is not a new idea to ameliorate video transmission, it is a hot topic today (as shown for example by the recent ISO MPEG DASH standard [23]). Many adaptation methods have been conceived. Some of them are sender-driven [28], others are receiver-driven [31]. The decision sometimes uses sender buffer [39], sometimes receiver buffer [28]. Some methods need beforehand data [13]; as a consequence, no method can deal optimally with all types of video transmission such as videosurveillance, videoconference, VoD (video on demand). There are a multitude of papers about this topic in the literature, however there is no article in the last ten years to classify the approaches used for adaptation.

In this context, this papers fills this gap by classifying adaptation methods using a taxonomy of the parameters used for adaptation decision. This allows to better understand adaptive video and have a global vision on how it can be done. A valuable and realistic adaptation method is afterwards suggested, without further evaluation.

The scope of this article is IP-based networks. Among the many steps in video transmission over IP-based networks where optimisation can take place, such as ALF (Application Level Framing), FEC (Forward Error Correction), video encoding and compression, this article focuses on the adaptation step. We only take into account solutions which do video adaptation and present their specification in scientific or white papers. Also, we consider in this article *only network parameters and network performance criteria*. Moreover, we focus on adaptation caused by network conditions, so we do not take into account other parameters, such as CPU load, available codecs, resolution and other terminal capabilities discovery at the beginning of communication on the client side.

Note that this article presents the network viewpoint. As such, papers which deal with image information results (such as image encoding, image quality, subjective or objective evaluation of received videos) without taking into account network viewpoint are out of the scope of the current article and have not been taken into account.

This paper is organised as follows. The next two sections present the positioning of current paper and its motivations. Section 4 presents various adaptation methods found in the literature. Sections 5 and 6 describe criteria used for the taxonomy, and respectively discuss the methods presented with respect to criteria proposed. Other surveys are given in section 7.

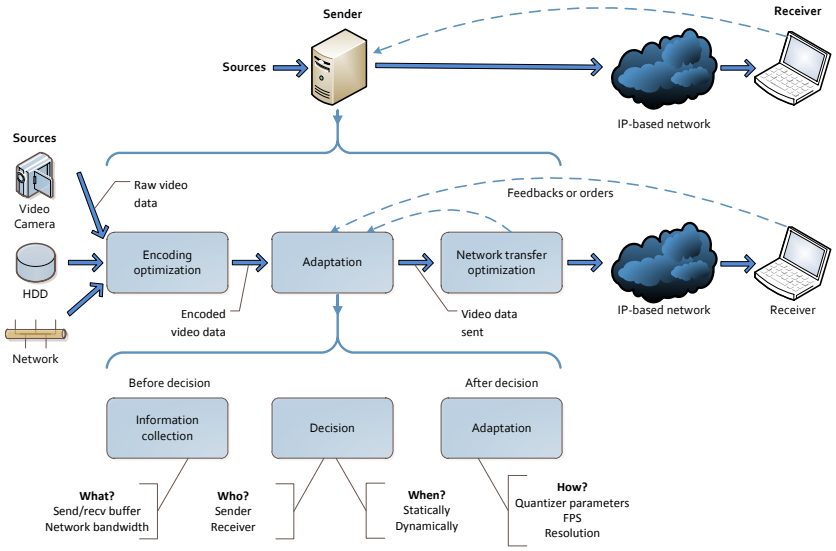

Fig. 1 Classical adaptive video transmission, the parts where transfer optimisation occur, and the details of adaptation process.

2 Context

The goal of the article is to classify adaptation methods from some points of view. This section details these points of view.

Fig. 1 presents a classical adaptive video transmission divided into three parts. The **upper part** of the image presents the classical figure of a closed-loop control between the sender and the receiver allowing to make video adaptation.

This is further detailed in the **middle part** of the image, which presents the video data transfer optimisations we have identified (we focus on end-to-end solutions, not on network ones, such as network resource reservation and service classes):

1. optimisation of encoding (video formats);
2. optimisation of adaptation;
3. optimisation of specific video transport protocols.

It should be noted that each of the three categories can have several optimisation methods. Also, a general video adaptation method can belong to several categories at the same time, i.e. it contains ideas which affect several parts of the transfer. Moreover, several methods can be used to further optimise video transmission.

The *first category* optimises specifically the video encoding in order to be the most adapted for the transfer over the network, and/or for other related purposes (such as storage on servers and encoder/decoder complexity). The signal processing community is constantly working to create new standards of video encoding, such as H.264 and its SVC extension, Theora and VP8, which increase the compression performance while maintaining a good video quality.

The methods in the *second category* act directly on the video data to be transmitted, for example by choosing the bitrate to encode the video. They can be divided in methods which add redundant information to video data, and methods which just change the bitrate of the video. For example, the former can use Forward Error Correction (FEC) [48] to protect the application against transmission errors at the expense of additional network resources used for this data.

The latter allows the flow to be network-friendly, and we will use the word "adaptation" for this. Adaptation can be done in three ways:

- Using stream switching techniques. The video is encoded in several streams (files) with different qualities (bitrates) and the adaptation method dynamically switches among the streams so that the bitrate matches the available bandwidth of the network.
- Using layer switching techniques. The video is encoded into several, potentially orthogonal, layers: a base layer and enhancement layers. The adaptation method sends the base layer and as many enhancement layers as possible depending on available bandwidth.
- Using direct encoding techniques. The adaptation method re-encodes periodically the transmitted video based on the available bandwidth.

A widely-used layer switching technique is scalable video coding (SVC), which provides temporal, spatial and quality scalability. Using this technique, the video is stored as a single stream (file), which contains several enhancement substreams of the three scalability types. The particularity is that if enhancement substreams are removed, the remaining bit stream is still decodable, hence making the original stream scalable. For our purposes, removing substreams means not sending them on the network because of the adaptation process, otherwise said decreasing video bitrate. A drawback is that an SVC stream is less efficient in coding than an optimised single-layer stream at the same bitrate [46]. An SVC technique for H.264/AVC with low efficiency loss has been standardised [46,52]. It leads to a high traffic variability [9], particularly suitable to adaptation.

Finally, the *third category* focuses on improving the network side of the transmission. Methods belonging to this category improve the way packets are sent without changing the video data itself. They act on many parameters. First, depending on the type of streaming, there are more appropriate transport protocols, such as DCCP (Datagram Congestion Control Protocol) [15]. Then, there are methods which selectively transmit packets (discarding some of them), based on available bandwidth [16,19], or selectively retransmit packets, based on their importance (I frames in an MPEG-encoded video are always retransmitted, contrary to P and B frames, for example) [22]. There are also methods that improve the performance of transport protocols on certain types of network so that the rate of transmission is maintained at an appropriate level [42]. Furthermore, a commonly-used optimisation technique is ALF (Application-Level Framing), which cuts intelligently video data in packets (avoiding for example to put one small video frame in two packets, which would be more sensible to packet loss); this is to be used in conjunction with the MTU (Maximum Transmission Unit) of the network.

The focus of this article is to present the adaptation process which is shown in the **lower part** of the image of the figure.

We first note that the adaptation is *not* needed in some particular cases, for example:

- If the network bandwidth is relatively stable or at least is known in advance, then a sufficiently big receiver buffer correlated to a not so small startup time on the receiver removes the need of adaptation. The server however has to smooth data sending in order to match the available bandwidth, especially when the video uses VBR (variable bit rate) encoding. Several smoothing algorithms have been proposed in the literature, with various complexities and properties [54, 34]. They have been compared in [14], and the implications of smoothing on bandwidth and delay analysed in [10].
- Contrary to video communication (which is the scope of the article), real-time audio communication uses a small bandwidth. Thus, the authors of [43] make the implicit assumption that audio data will not exceed the available bandwidth and will not be lost. Hence, only the *delay variation*, given by RTT, needs to be addressed. They propose an ingenious mechanism to playout delayed packets, described in the following. The receiver uses a small buffer. It is assumed that audio data contains voice periods and silence periods. When voice data arrives in late, due to increased RTT, the receiver adds silence periods to a currently-played silence period. When RTT decreases back to normal, voice data arrives faster than its playout, and the receiver cuts from silence periods, thus restoring the normal delay. To resume, the client hears the same voice data, but with silence periods shortly longer or shorter.
  It should be noted that in video case there are no "silence" periods which can be shortened or lengthened. Additionally, the available bandwidth may not be sufficient, and considering the RTT only is not good enough: if bandwidth becomes too small, packets get accumulated in sender or network buffers very fast and eventually are dropped there. Thus, in video case the problem is not only with the delay, but also with data packets which get lost.

In the general case where the adaptation process is used, it needs a decision about bitrate increasing, decreasing or maintaining. The adaptation process can thus be divided in three time frames: before, at and after adaptation decision. In the first time frame, all the necessary parameters needed by the decision are gathered. The decision is then taken at the second frame. Finally, once the decision has been made, in the third time frame the adaptation itself can take place. The three time frames are analysed below.

In the "before decision" time frame, several parameters exist to help taking the adaptation decision. For example, the available network bandwidth or the filling level of the receiver buffer. We group them in the *what* group, which is the main group in this time frame.

At the moment of decision, we take into account *who* does the decision (sender, receiver) and *when* it does it (e.g. each second). The decision is whether the bitrate should be increased, decreased or maintained.

Finally, after the decision, several parameters, grouped in the *how* group, can be changed. For example, modifying the bitrate can be done by modifying the quantizer parameter of the codec, or modifying the number of FPS (frames per second) or the resolution or other parameter, or changing the level in a multi-level encoded video. This can also take into account the client characteristics, for example the maximum resolution on that particular device or the set of video codecs supported.

Video adaptation is a multidisciplinary domain. The network research community pays more attention to the first time frame (information gathering about transfer speed) and the second time frame (adaptation decision), and the image research community is especially involved in the third time frame (video quality modification). The time frames can be optimised independently of each other. *This article focuses on the first two time frames, i.e. it does not present how video is changed to match the desired bitrate.*

3 Complexity of adaptive video transfer

Adaptive video transfer is more complicated to understand than it appears. One of the reasons is that different speeds are involved. As is shown in Fig. 2, several parts affect the speed of a communication:

- the sender: the application generates data at some speed (i.e. generating bitrate), and the transport layer may modify it by buffering and delaying it (i.e. sending speed);
- the network has its own speed, given by its resources and their instantaneous usage (i.e. data transmission speed);
- the receiver: the transport and/or application layer buffers data so that it is shown timely on the user screen (i.e. data consumption speed).

We analyse the variation of these speeds (i.e. whether they are constant or dynamic) in the following.

We assume that network congestion control is used. We consider that the amount of data can be measured either in bytes, or in video duration (seconds of video). During each second of the transmission:

- the sender application sends an amount of data either in terms of bytes or of seconds. It can send either a *constant* duration (one second of video, in a videoconferencing/videosurveillance case), or a *dynamic* duration (for example several seconds of data in a VoD case). On the other hand, it generally sends a *dynamic* amount of data in bytes (equal to current bitrate, which changes over time);
- similarly, the network transports a *dynamic* amount of data (either in bytes or in seconds);
- the receiver application consumes a *constant* video duration of data (one second of video), and a *dynamic* amount of bytes of data (equal to sender bitrate in an ideal network).

On the other hand, the sender has control over its own dynamicity, since it can change the bitrate. The video system (sender and receiver) has no control over network speed

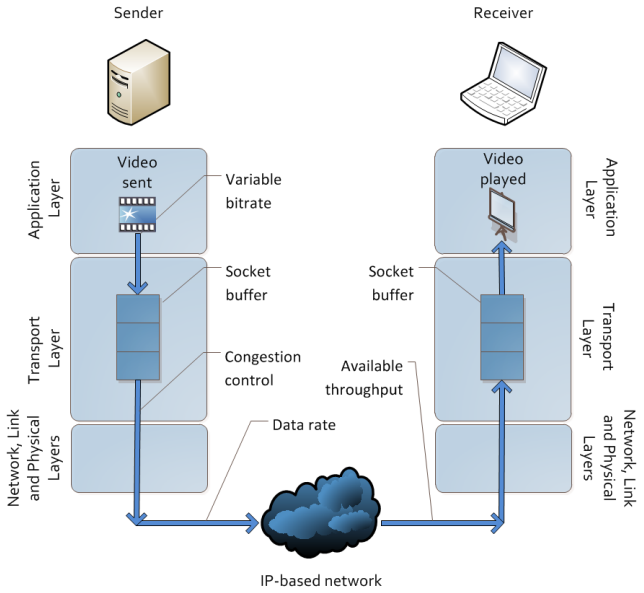

Fig. 2 The various speeds involved in a video transfer.

	Sender app	Network	Receiver app
Secs	const or dyn (ctrl)	dyn (non ctrl)	const
Bytes	dyn (ctrl)	dyn (non ctrl)	dyn (sender&net-ctrl)

Table 1 The variation and controllability of the speeds involved in a video transfer.

since it is defined by available bandwidth, an external factor to the video system; the network is also responsible for lost packets; thus, network input (at sender) is not necessarily equal to network output (at receiver). The receiver consumes an amount in bytes at the speed given by the sender bitrate and network bandwidth, while the data duration consumed is constant (except if the user pauses the video stream). So the only uncontrollable factor comes from the network. This is summarised in Table 1.

Given the discussion above, we end up with three different speeds (on sender, network and receiver) and two different measurements of speed (in duration or in bytes). All these parameters are correlated, for example when network speed increases, receiver data accumulates and sender buffers become empty, hence the sender speed can increase. As such, adaptation methods can use any of them, and that is the main reason why several types of methods exist. Next section presents some of them.

4 Methods presentation

In this section we present various video adaptation methods found in the literature with regard to the parameters used by adaptation.

4.1 Methods using information from sender buffer

4.1.1 VAAL, Video Adaptation at Application Layer

VAAL [39,41] is a video adaptation method applied on the server side which does not require modification of the client. It uses transport protocol buffer overflow to find out the available bandwidth and to adapt video bitrate to the discovered bandwidth. Each n fixed seconds (or each GOP, Group of Pictures, for example), the server application computes WFP (Write Failure Percentage) of the packets which failed to be written to socket buffer. This number is used to control the video bitrate afterwards. A high number means small bandwidth, hence the bitrate needs to be reduced. Zero error indicates either a stable or more bandwidth, so the bitrate of the sent video can be increased. In this way, the bitrate of the watched video is fixed during each period of n seconds, and can change only between periods. VAAL uses the value of WFP to decide whether to increase video quality ($WFP = 0$), to maintain video quality ($0 < WFP \leq 5\%$), or to decrease video quality ($WFP > 5\%$).

4.1.2 QAC, Quality Adaptation Controller

The Quality Adaptive Controller QAC [8] uses feedback control theory for video streaming. The adaptation of the video is done on the server side, which stores the same video in several files with different bitrates. The client only supports the decoding of the received video.

QAC uses HTTP and an application buffer. A worker thread continuously takes video packets from the application buffer and sends them to TCP. It monitors the size q of the application buffer. It chooses the highest bitrate which makes q greater than a predefined threshold q_t, while keeping at the same time the data rate under available bandwidth. So, the buffer has always data to be transmitted. The available bandwidth in QAC is modelled as a disturbance, as explained in [32].

4.2 Methods using information from receiver buffer

4.2.1 Buffer-driven

The Buffer-driven adaptation method, described in [28], takes into account the occupancy of the receiver buffer to infer which video quality should be chosen and streamed. The occupancy B_E of the receiver buffer is used at the sender by the following equation (presented in [27]):

$$B_E = B_C + \left(\frac{pktnum * s}{i} - ER \right) * RTT \tag{1}$$

where B_C is the last buffer occupancy, $pktnum$ the number of packets, s the packet size, i RTCP (RTP Control Protocol) interval, and ER the encoding rate.

This method has also its own mechanism to calculate the sending rate R. R value is used to calculate two thresholds th_min and th_max. If the occupancy of the receiver

buffer is between 0 and th_min the video quality is reduced in order to avoid under-utilisation of memory. On the other hand, if it is between th_max and 100% video quality is increased to prevent over-utilisation of the memory.

Another adaptation mechanism based on the occupancy of the receiver buffer is described in [55]. This mechanism, contrary to buffer-driven, does not take into account the stability and the fairness of the network because it uses UDP as a transport protocol for video streaming.

4.2.2 MVCBF, Multiple Virtual Client Buffer Feedback

MVCBF [30] is defined as a new type of RTCP feedback at the application level. It aims to improve the information messages between the client and the server in a video streaming based on the SVC extension of H.264/AVC. For that, the proposed solution defines a virtual buffer for each layer of the video transmitted to the client to store its data. Then the client sends MVCBF feedbacks for each virtual buffer VCB (Virtual Client Buffer). After receiving a feedback the media time MT for each layer is calculated, then it is compared to a predefined one out of 5 thresholds ($T1$ to $T5$) in order to determine the appropriate adaptation action.

- If $MT > T5$ add a layer.
- If $T2 < MT < T3$ maintain the video quality at the current layer.
- If $T1 < MT < T2$ increase the number of frames per second.
- If $MT < T1$ reject a layer of the transmitted video.

The threshold $T4$ is used to control the time of the base layer. If $MT < T4$ for this layer, the application must reject directly the last added layer. On the other hand, if there has been no MVCBF feedback for a while, a new layer is added.

4.2.3 AHDVS, Akamai HD Video Streaming

AHDVS [7] presents the method Akamai CDN (Content Delivery Network) uses for video adaptation, which employs HTTP connections. Videos are encoded at five levels with H.264 and spacing time of 1.2s between two I keyframes, which means that adaptation is performed every 1.2s. The client provides regular information to the server needed for transmission and especially the adaptation algorithm, such as the bandwidth estimated by the receiver, the size of the receive buffer, the number of frames received per second and current bitrate received.

To decide what quality to transmit (the data rate at which the application should supply the TCP buffer), the algorithm uses the receiver buffer size q and a predefined threshold q_t. The receiver looks regularly at the variation of the receiver buffer (filled size) and tries to maintain it constant. If q is equal to q_t, the algorithm maintains the quality. The quality is decreased when $q_t - q$ increases. Otherwise, the quality is increased.

4.3 Methods using information from network

4.3.1 Kofler et al.

Kofler *et al.* [25] present a method that handles the streaming towards a wireless terminal. The adaptation method presented uses a platform for streaming based on MPEG-21 [4] to stream multi-layer videos based on SVC extension of H.264/AVC. It is applied on the server side and operates on two layers, application and transport layers, but other layers, such as the physical layer, can provide additional information. The video is divided in several units, using the generic bitstream syntax decription (gBSD) tool. The adaptation is done on a unit as a whole. This technique is similar to dividing the video in several independent parts. The adapted units will be later sent through the RTP protocol.

For network bandwidth estimation the authors also implemented the equation used by TFRC flow while using a feedback system based on RTSP to deliver network information (input parameters of the equation of TFRC throughput) from the client to the server. This information is sent periodically (once per RTT, Round-Trip Time).

4.3.2 Eberhard et al.

Eberhard *et al.* [12] also present a platform based on the MPEG-21 format and the SVC extension of H.264/AVC for adaptive streaming of video on demand (VoD) and for multicast. For VoD, the adaptation unit ADTE (Adaptation Decision Taking Engine) has been integrated into the server using the *vlc* video player.

Having received HTTP requests from the client, the server responds by sending a list of available sequences. The user selects the sequence and specifies the adaptation parameters which will be sent to the server, through HTTP again, using the MPEG-21 description. At that time the server uses RTP/RTCP for transmitting the sequence requested. A description for each media to be transmitted is created which specifies the parts to keep and those that can be rejected or modified. During transmission, this description is used to make the adjustment based on the client's environment.

4.3.3 Wien et al.

The authors of [51] also create a streaming architecture based on the MPEG-21 and videos encoded in SVC of H.264/AVC. In this architecture the static information and preferences of each client are first collected and sent to a special node, which is responsible for the adaptation of the video. When a user requests a video, a representation of MPEG-21 is made to adapt the transmitted video to his profile. After preparing the MPEG-21 representation, the application sends the video to the receiver via another node responsible for the adaptation.

If during the transmission the context changes, such as the available bandwidth or performance of the video player, adaptation node is informed. This allows to readjust the parameters used for the current transmission such as allocating more or less bandwidth for each media transmitted. If however changes are severe, the adaptation

node can reconfigure the representation of transmission by sending a different format of the video, a still image or even only the soundtrack of the video requested.

4.3.4 VTP, Video Transport Protocol

Video Transport Protocol (VTP) [3] is a protocol at the application level created by the authors themselves. It was specifically designed to transmit videos encoded in MPEG-4. The main goal of VTP is to minimize the loss of I key images of sent video by sending fewer packets when the network is congested, and therefore having fewer I-frames deleted upon receipt. To do this VTP acts on two factors: video bitrate and sending rate. The video should be previously encoded in several bitrates. The sending rate is calculated at the sender, through a special interaction with the client which regularly sends acknowledgements at each predefined interval. Thus, VTP is based on transmission rate and not on congestion window like for TCP.

VTP controls the transmission every k packets. The transmitter sends a request for acknowledgements and receives replies. This exchange allows it to calculate the RTT, the sending rate, and subsequently the best bitrate to be used. To calculate the available bandwidth, VTP applies an EWMA-based algorithm on many previous adaptation periods.

The method described in [24] acts the same way as VTP, but it creates a protocol at transport level.

4.3.5 Gorkemli et al.

Gorkemli *et al.* [18] create an adaptive platform to transmit multi-layer videos encoded with SVC extension of H.264/AVC. The quality of video transmitted is determined by the conditions of the network via DCCP or TCP. The sender estimates the sending rate on the network, extracts a group of images (GOP) from the video so that its rate matches the estimated available bandwidth, and finally sends the extracted video packets to the receiver. As packets arrive at the receiver, they are inserted into a special buffer for the decoder, large enough to filter out variations in the network. The decoding starts when half of the buffer is full.

Since the enhancement layer needs the base layer to be decoded, the application uses an adaptive scheme of automatic repeat request (ARQ) (Automatic Repeat reQuest) to reduce traffic retransmission. This scheme only requires the retransmission of lost packets on the network belonging to the base layer when the network capacity is low, and requests retransmission of all missing packets when the network allows.

The adjustment method is described in a later article [17]. The rate controller module regularly computes the sending rate. Each fixed T_{rc} milliseconds, the sender computes the average number of bytes deposited to transport protocol. The average is computed for the interval $T_{current} - T_{average}$ and $T_{current}$, where $T_{average}$ is fixed. The layer extractor module dynamically chooses the extraction rate at sender: When the number of NAL units in the sender buffer decreases below a fixed QS_{min} value, a new extraction rate is computed. The new extraction rate is basically the current sending rate plus the minimum between zero and the difference between sending and current extracting rate.

4.3.6 MPEG-TFRCP

MPEG-TFRCP (MPEG-TCP-Friendly Rate Control Protocol) [50] re-encodes the video on the fly to adapt the bitrate of the sent video to the network available bandwidth. The adaptation is done by changing the quantization parameter of the encoder. The authors propose to make the adjustment each period of 32 $\times$ RTT. The available bandwidth r_i used for adaptation is a modification of TFRC's one:
if $p > 0$

$$r_i = \frac{MTU}{RTT\sqrt{\frac{2p}{3}} + T_0 \min(1, 3\sqrt{\frac{3p}{8}})p(1+32p^2)} \tag{2}$$

else

$$r_i = 2 * r_{i-1} \tag{3}$$

where T_0 the retransmission timeout, p the packet loss probability and MTU is the Maximum Transmission Unit.

4.3.7 Evalvid-RA

Evalvid-RA [29] is a framework enabling *simulation* of rate adaptive videos. The framework uses the classical ns2 network simulator, real videos as input and generates trace files on receiver which can be assembled to form a video as received by the client. This allows not only to compute a subjective quality of the received video, but also to compute in a reproducible manner the PSNR of the received video. It supports H.264 as video codec, and DCCP/TFRC as transport protocol. The quantizer parameter is changed in real-time in order to adjust the sending rate according to bandwidth capacity. The latter information is provided by TFRC congestion control.

4.3.8 CBVA

CBVA is an adaptation method for streaming of stored video files at the sender side [13]. CBVA encodes each video in multiples qualities (different bitrates and frame rates) and for each one it creates in advance a group of operating points (combination of frame rate and frame quality). When available bandwidth changes, CBVA changes the streamed video to another operating point. For this, there is a deadline for each GOP of each operating point. Video quality is increased if this deadline is highly respected, i.e. the GOP time is in advance to its deadline. On the contrary, if the deadline is exceeded, the video quality is decreased.

4.3.9 DRDOBS, Delay-Aware Rate Distortion Optimized Bitstream Switching

DRDOBS is a video adaptation method based on the transmission delay [53]. It uses a virtual buffer approach to estimate this delay, and to avoid packet loss and packets re-caching. The virtual buffer is placed in the network between the server and the

client. For a streaming application the buffer capacity B at time i should satisfy the following equation:

$$0 \leq B(i) \leq \sum_{j=i+1}^{i+N_t} C(j) \tag{4}$$

where N_t is the interval for decoding the following N pictures and $C(j)$ represents the network capacity. The rate of sending and receiving are assumed to be equal.

DRDOBS proposes to change the quality between two bitrates. The highest bitrate is used while the above equation is not violated thereby avoiding under-utilization of the receive buffer. Otherwise, the algorithm switches to lower bitrates.

4.3.10 Schierl et al.

Schierl and Wiegand [45] propose an adaptation method for videos encoded with H.264/AVC. Their method combines a technique of temporal scalability of the video and a flow switching technique based on the available bandwidth. The first technique is used to reject enhancement layers in response to a reduction in available bandwidth; as this article was written before the apparition of the SVC extension of H.264/AVC, the scalability used is a simple temporal scalability. The video is divided in three layers: an independent base layer with only I images, a first enhancement layer using only P images (which depend only on the base layer), and a second enhancement layer containing B images (which depend on both previous layers). In the second technique, the video is encoded with multiple values of QP (Quantization Parameter) which results in a video with various bitrates available on the server. Depending on the available bandwidth, which is calculated using an algorithm of type IIAD (Inverse Increase Decrease Additive), the adaptation method (implemented as a modified version of RTP/RTCP) switches between available bitrates as follows:

1. If $\Delta < s1$, use flow switching.
2. If $s1 < \Delta < s2$, use temporal scalability and drop highest enhancement layer.
3. If $s2 < \Delta < s3$, use temporal scalability and drop both enhancement layers.
4. If $\Delta > s3$, use flow switching.

where $s1 = 500ms$, $s2 = 1000ms$, $s3 = 1500ms$ and Δ the packets receiving delay.

4.3.11 RAAHS, Rate Adaptation Algorithm for HTTP Streaming

RAAHS [31] proposes an adaptation algorithm for adaptive video streaming over HTTP. This algorithm is executed on client side and uses the loading time of a video segment (SFT, Segment Fetch Time) to detect congestion and calculate the transmission rate over HTTP. The video quality is controlled at the user by a method which increases it by one step and decreases it significantly in case of congestion. For that, the sender sends video data divided into segments of MSD (Media Segment Duration) seconds which have values between 5 to 10 seconds, and the receiver looks at the SFT time of each segment. Then, it compares the SFT with the segment duration MSD to decide whether to increase or to decrease the video quality. If the loading time SFT of a segment is greater than its duration, RAAHS considers that TCP throughput is

lower than the bitrate of the video and the quality is decreased. Otherwise, it considers that the available bandwidth is greater than the bitrate and the quality is increased.

4.4 Methods using information from sender/receiver buffer and network

4.4.1 Nguyen et al.

The method described in [33] creates a new congestion control mechanism to better adapt the streaming of videos encoded using SVC extension of H.264/AVC. The video is divided in several spatial layers, several temporal layers, and several quality layers with one base layer and several quality enhancement layers.

The available bandwidth is estimated in two different manners. At the beginning of the transmission, the available bandwidth is estimated through a technique similar to PTR (Packet Transmission Rate) [21], which is a variation of the well-known *packet pair* technique. Afterwards, the client periodically sends to the server the estimated bandwidth B, the occupancy of the receiver buffer and acknowledgements for base layer packets. Packets belonging to the base layer have a higher retransmission priority compared to packets for the other quality layers. Based on this information, the server decides which layer to send to the client. If B is higher than the data rate of current spatial and temporal resolutions, then it either increases the temporal resolution if B is less than the next higher spatial resolution, or increases the spatial resolution otherwise. Elsewhere, it reduces the spatial and temporal resolutions. The difference between B and the current sending rate is filled with as many quality enhancement layers as possible, from lowest to highest temporal layer. This is possible because enhancement layers, coded using fine-grained scalability (FGS) through a technique similar to sub-bitplane coding, can be truncated at any point without affecting the decoding process [33].

4.4.2 Methods proposed by major companies

We present here some well-known solutions proposed by major current companies and a currently developing standard. They are all based on HTTP over TCP, and have some other common points. In all of them, the video is encoded in different bitrates and the resulting files are divided (either virtually upon client request or at content creation) to multiple independently decodable chunks, for example each interval of k seconds of the video (0-k, k-2k, 2k-3k, 3k-4k etc.) are stored in M files with M different bitrates. The general principle is that the client initially fetches a manifest file with information about available files (such as quality, duration and file name) and afterwards it regularly measures when data of the k seconds arrived to infer network capacity; for example, if it arrived in less than k seconds, then the network is faster than the current playback, hence for the next k seconds the client requests a file with higher bitrate. These requests use standard HTTP GET transactions. In the following we will present some additional information.

IIS Smooth Streaming is a web-based live adaptive streaming service provided by Microsoft [56]. This service uses a plug-in that is available for Windows and iPhone OS 3.0. This plug-in is in fact a codec which performs a video stream-switching approach. The video is available in different bitrates, about 7 ranging from 300 kbps up to 2.4 Mbps, and resolutions up to 1080p.

Adobe HTTP Dynamic Streaming [11] is also a web-based live adaptive streaming service but it is developed by Adobe. It is available for all devices which have a browser with Adobe Flash plug-in. The server stores multiple streams with different resolutions and qualities. The service alternates between the different qualities during playback so that to match the available bandwidth of the users and their CPU capacities. Supported video codecs include H.264 and VP6, which are currently included in the Adobe Flash plug-in.

HTTP Adaptive Live Streaming is an HTTP client for adaptive streaming released by Apple [35]. The server divides the video content into several pieces with many configurable qualities and durations. The server proposes a playlist (using M3U8 extension) containing all the available video segments. The client downloads dynamically the segments of the video which match the available bandwidth using an undisclosed algorithm. The algorithm uses the H.264 codec together with the MPEG2 TS container. This service is available for any device running the iPhone OS 3.0 or later (including the iPad), or for any computer with QuickTime X or later version installed.

MPEG DASH (Dynamic Adaptive Streaming over HTTP) [23,47] is a recent ISO standard technically similar to the three HTTP-based methods above, but vendor-neutral and audio/video codec-neutral. The adaptation is driven solely by receiver, and can be based not only on network resources, but also on device capabilities and user preferences. It should be noted that MPEG DASH defines the manifest file, the video segment formats and quality metrics of the transmission, but does not specify precise adaptation heuristics. Several DASH server and clients exist, both free software (vlc) and proprietary software, and on various platforms (GNU/Linux, Windows, OSX, Android, iOS and others).

5 Taxonomy criteria description

After reviewing several articles on video adaptation, we focus now on creating a taxonomy for adaptation methods. This section details the criteria used for the taxonomy. Some criteria precise the context where the method can be used, an information especially useful to the video provider. Others precise the parameters needed for the adaptation method (how the adaptation process works). Finally, each method presents some specific characteristics. Therefore, we divide criteria in three parts: context, functioning and properties.

Note that we take into account criteria related to the *adaptation idea*. We are not interested, for example, if the method was validated through simulations or real experiments, or if the examples given in the article were for videoconference or for VoD

or for another video type. Moreover, we focus on before adaptation and at adaptation time frames, not on after adaptation.

5.1 Usage context

Compatibility with ahead of time (beforehand) data availability: whether the method works when no or few data is available on the sender before transmission. Based on the amount of data available (or data generation) before adaptation or sending process, we identify three cases:

1. no data is available: this is the case for videoconferencing and for critical videosurveillance (CCTV, closed-circuit television), where data must be sent immediately after its generation.
2. some data is available: this is the case for digital broadcasting television, where data is sometimes delayed by a few seconds (the so-called seven-second delay or broadcast delay), and for non-critical videosurveillance.
3. all the data is available: this is the case for VoD (Video on Demand) or files stored on server.

The next table presents an example of this criterion on three hypothetical methods:

	Method 1	Method 2	Method 3
No data	yes	no	no
Few data	yes	no	yes
All data	yes	yes	yes

It is useful to be sure that a method which needs no data for example works also when few or all data is available. The following theorem treats this.

Theorem 1 *If a method works when n seconds of data is available, then it also works when N seconds of data ($N > n$) is available. In the previous table, this means that if an adaptation method works (has "yes") on one line, then it works for all the lines below it.*

Note: Reading the data poses no problem, since we can easily consider that the method which works with n seconds should just read n seconds instead of the N seconds available. As usually, the problem arises when the method wants to *write* data, i.e. to alter data, for example it wants to re-encode it to a different bitrate, because the data has already been generated.

Proof: Suppose a method which works when n seconds of data is available beforehand. Suppose that an intermediate machine is added between the data generator and adaptation method, whose role is to delay the video by $N-n$ seconds. When data generated needs to be altered, the intermediary is contacted instead of the generator. Thus, the data is generated N seconds before, and is seen by the adaptation method n seconds before. Q. e. d.

Note: Adding an intermediate machine is useful to prove the theorem. Of course, using an intermediate machine could not be the best solution for a given method. For example, for methods which can re-encode the n seconds themselves, there is no need of an intermediate machine.

Usage: We consider here whether the method is appropriate to real-time video transmission (such as videoconferencing) or to delay-tolerant transmission (such as video on demand). A "real-time" method is of course appropriate for delay-tolerant transmission too. Also, if a method needs some beforehand data, then it cannot be real-time.

5.2 Functioning

Who: who takes the decision of the adaptation (whether it should increase, decrease or maintain video quality).

This should not be confounded with who does the adaptation, which is always the sender, since the video is found on sender (for our purposes, adaptation on an intermediate network equipment, i.e. proxies, are included in sender part). It should not be confounded with who makes calculus either; the calculus can be made by several parties (for ex. sender computes the bandwidth and the receiver computes the packet delay), but only one party takes the decision. We do not take into account who makes calculus, since as far as we have seen calculus are simple and fast (such as summing up several values) compared to when the decision is made.

Two values are possible for this parameter: sender and receiver.

What: what information (parameters) are used to take the decision. Two values have been found in the literature: buffer (at sender or at receiver) and network. Both can be measured either in bytes, or in seconds of video.

When: when the adaptation decision (or parameter evaluation) is performed (note that the decision could be to change one or several video parameters, or simply not to change anything).

Computing interval of parameters: what interval in time is taken into account for the adaptation compared to the adaptation interval. Suppose the adaptation decision is taken each N seconds. The decision could use the parameters from the whole interval, a smaller one (for ex. last RTT), or even the instant value of the parameter. For example, each 2 seconds the average data rate on the last RTT is used to choose the bitrate for the following interval of 2 seconds.

5.3 Properties

The functioning of a method can lead to specific properties. For example, some methods use the well-established HTTP protocol, others work with any transport protocol with congestion control. Some methods work with with multi-layered video only, while others work no matter the video encoding. Some methods support MPEG-21, which is a standard allowing, among others, the sender to find out the characteristics of the client (codecs available, maximum resolution allowed etc.)

6 Discussion

Table 2 presents a classification of the methods presented in Section 4 based on the criteria specified in Section 5. We discuss the results of each column, and afterwards make some general remarks.

Beforehand: In the beforehand column it can be seen that all the possibilities are present, i.e. some methods need no beforehand data, others need a few only, whereas other need all the video stream before starting transmission. As an example, methods with "all" work only in case of stored files (VoD), in which case they potentially give better results.

It is worthwhile to note that major companies solutions need all video data beforehand. Indeed, the manuscript file, which gathers information about all available video streams and allows client to regularly request the appropriate stream, is downloaded by client at the beginning of communication.

Usage: Some interesting points can be remarked. First, both values (real-time and delay) are found abundantly in the literature. Also, as written before, methods working in real-time also work in delay-tolerant transmissions.

Furthermore, methods based on TCP (such as [8, 13]) or which need few or all beforehand data (such as [12, 31]) are not considered to be real-time. Other methods, such as [51], use client-aware intermediary nodes for performing adaptation, and as such they are useful only for delayed video transmission. Some others, such as [53], use virtual buffers, a kind of aggregated buffer for buffers on server and intermediate nodes, which reduce lost packets rate but add delay.

We also consider that since all the methods presented adapt video, the moment when they adapt is not so important, i.e. adapting each RTCP interval or each 2 seconds does not prevent them to be real-time. We do not consider multi-layer encoding to have an influence on real-timeliness of a method either.

A final interesting remark is that major companies solutions need beforehand data, and as such they are not appropriate for real-time communications.

Who: In the table it can be seen that generally the decision is taken by the sender. This is not surprising, since most of the information used for the decision is found on the sender: the sender does the adaptation, it has the video data and information about them (available qualities/bitrates, size, encoding, future characteristics etc.) However, the solutions proposed by major companies, based on HTTP, are receiver-driven. In this case, the sender sends information about videos to the receiver at the beginning of the communication (through a manifest file), and the receiver regularly sends adaptation information to the sender (through HTTP requests).

The difference between research community, where the decision is taken by server, and industry, where the decision is taken by receiver, is interesting and can be explained as following.

First, as already said, video is found or generated on server, hence all the information about video is found on server. Taking the decision on server avoids such

	Context		Functioning				Properties
	Beforehand data needed	Usage	Who	What	When	Interval	
VAAL [39]	no	real-time	snd	snd buf, bytes	each *n* sec. (2 for ex.)	last RTT or 20 last pkts	any TP with CC
QAC [8]	no	delay	snd	snd buf, bytes	when buf exceeds thresholds	instant	HTTP
Buffer-driven [28]	no	real-time	snd	rcv buf	each RTCP interval	instant	RTP, RTCP
MVCBF [30]	few	delay	snd	rcv buf, secs	when feedback arrives	whole interval	virtual buff, ML, RTCP
AHDVS [7]	few	delay	rcv	rcv buf, bytes	each 1.2 sec.	N/A	HTTP
Kofler [25]	no	real-time	snd	net, bytes	when feedback arrives	RTT	TFRC, RTP, ML, MPEG-21
Eberhard [12]	all	delay	snd	net, bytes	at RTSP feedback	N/A	RTP, RTSP, ML, MPEG-21
Wien [51]	all	delay	snd	net, bytes	when feedback arrives	N/A	RTP, ML
VTP [3]	all	delay	snd	net, bytes	each *k* pkts	many previous periods	new transport protocol
Gorkemli [18, 17]	no	real-time	snd	net, bytes	when buf exceeds thresholds	a fixed interval	DCCP, ML
MPEG-TFRCP [50]	no	real-time	snd	net, bytes	each 32*RTT	whole interval	direct encoding
Evalvid-RA [29]	no	real-time	snd	net, bytes	each GOP	instant	DCCP TFRC
CBVA [13]	all	delay	snd	net, secs	each GOP	instant	TCP
DRDOBS [53]	few	delay	snd	net, secs	when buf exceeds thresholds	instant	virtual buff
Schierl [45]	no	delay	snd	net, secs	each RTCP feedback	instant	RTP/RTCP, ML
RAAHS [31]	few	delay	rcv	net, secs	each segment received	whole interval	HTTP
Nguyen [33]	no	real-time	snd	net, bytes & rcv buf	each GOP	instant	ML
Major companies [56,11,35,23]	all	delay	rcv	net, secs & rcv buf	each packet received	whole interval/instant	HTTP

Table 2 Classification of major adaptation methods found in the literature, grouped by *what* parameter.

information to be regularly exchanged between server and client, so this appears as a natural and optimised choice in research community.

Second, industry is bound HTTP/TCP, and there are several reasons for this. As industry *deploys* their solutions at *very large scale*, practical concerns/constraints are extremely important (such as has *to work, now* on Internet). Avoiding interconnection issues allows major companies to attain as much clients as possible. Nowadays HTTP, which is on top of TCP, is the works-everywhere application protocol (it is allowed in all networks: Internet, telecommunication networks).

Furthermore, solutions based on HTTP and with client-side decisions scale well: they can use HTTP caches already in place (because data for a given address does not change) and current content delivery networks (CDN), they yield a stateless protocol (the server does not keep track of clients), work well when client pauses/resumes the video (since the video is not unnecessarily transmitted during pause), do not overwhelm clients which do not have enough CPU power to process all data received from network, they work in IP multicast or with many receivers of the same video source etc.

It should be noted that HTTP by itself does not prevent the server to take the decision, as shown by QAC [8], where the decision is taken by HTTP server. However, this means that all the advantages on scalability presented above disappear. Also, this means that the server needs to be modified in order to check current network conditions. In QAC for example, the HTTP server fills an application buffer and a worker thread drains it by sending packets the TCP. As a side note, this also adds a small delay to transmission.

What: The *what* parameter is the essential and the most complex parameter. Indeed, to be able to know whether to increase, maintain or decrease the video quality, the decider needs some information about the transmission speed. Feedback from receiver is required, usually through a transport protocol with congestion control or through an application protocol such as RTCP or HTTP. We have identified in the literature two sources to get this information: from sender or receiver buffer, and from network. Both can be viewed in terms of bytes, or of seconds of video. They are explained in the following.

The first source of information is thus sender or receiver buffer. When the network is too slow, packets get accumulated on sender buffer. When it is too fast, packets get accumulated on receiver buffer. Thus, the filling level of those two buffers of transport protocols (also called *sockets* in programming languages) expresses transmission speed. Note that the use of these two buffers is not a new idea, since they are very similar to congestion window and receiver window used in classical congestion control and respectively flow control of TCP transport protocol.

The buffer can be measured in two distinct ways: either in *bytes*, or in *seconds* of video. For example, the same size of data in buffer, 1Mb, corresponds to 1 second of 1Mb/s video, and to 4 seconds of 256kb/s video. When buffer is measured in bytes, methods can look at the filling level of the buffer, or only whether the buffer is completely filled or not.

Classically, the adaptation using buffer information is done as following. On the sender side, a filling buffer (either in bytes or in seconds of video) means a network

speed slower than data generation, hence the application should generate less data, so it should decrease video quality; and it should increase quality for buffer lowering. Conversely, on the receiver side a filling buffer (either in bytes or in seconds of video) means too much data arrived, hence network is faster than data consumption, hence sender can send more data by increasing video quality; a lowering buffer should cause a quality decrease so that an empty buffer (leading eventually to video freezing) be avoided.

From an implementation point of view, buffer filling information is currently provided by some operating systems and only for TCP[1]. Otherwise, the sender can "simulate" it by using its own application buffer as following: The sender puts packets in an application buffer, while another thread continuously moves data from this buffer to transport protocol buffer; thus the application has access to the filling level of application buffer.

The second source of information is the network itself: its speed, its delay or other related parameter. For example, when the network speed is slower than bitrate, or when N seconds of video are received in $M > N$ seconds, packets do not arrive in time to receiver; when the network is faster than bitrate, the quality can be increased.

Again, the network information is divided in *bytes*, for example the available bandwidth or throughput, and *duration*, for example the time taken by some data to be transferred on the network between sender and receiver.

Network information can be obtained from transport protocols with congestion control, such as TCP ([13]) or DCCP/TFRC ([25]), or otherwise from application protocols, such as the classical RTP/RTCP ([52]). In the former case, transport protocols estimate bandwidth (or other network-specific parameter, such as instantaneous TCP congestion window size) which can be read at application level[2]. In the latter case, regular control feedback is sent from receiver to sender with information such as number of lost packets.

To resume, the sender application should decrease bitrate when the sender buffer fills up or when the network bandwidth lowers or when the receiver buffer lowers.

It is interesting to see what happens when the user stops the video for some amount of time (5 minutes for example). Methods which use the receiver buffer will notice this, by seeing the buffer filling up, hence they can for example increase quality until the best one and keep it sending. The other methods act as if nothing happened, filling up the receiver buffer. Nevertheless, this is subject to transport protocol constraints on receiver buffer, such as flow control in TCP.

It should be noted that the change in bitrate could be subject to constraints other than network-based. For example, VAAL [41] includes an oscillation-avoiding algorithm which prevents bitrate increasing if it leads to quality oscillations, and DR-DOBS [53] uses rate-distortion optimisation, where the bitrate is changed only if the quality increasing is worth the extra bits. We do not detail such parameters because, as written in the introduction section, this article focuses on network parameters.

[1] On GNU/Linux, a `tcp_info` structure with such information is returned by `getsockopt` function with `TCP_INFO` as parameter.

[2] See the previous footnote, on TCP.

It is still an open question which *what* parameter is qualitatively better. Some methods (Nguyen [33] and HTTP-based [56,11,35,23]) even use two parameters: receiver buffer and network. However, these parameters, either in bytes or in seconds of video, are dependent each other, which means that their combined usage does not give much more information than one parameter alone. For example, the fact that 2 seconds of video data have been received in 1 second can be measured in two distinct manners: measuring how much data has been transmitted (from the network perspective), and how much data the buffer has received (from the buffer perspective). Also, it is known that video bitrate equals data size divided by video duration, and network throughput equals data size divided by transfer time. This means that if in 2 seconds the receiver buffer received 1Mb of new data (buffer measurement), then the network throughput was 512kb/s (network measurement).

When: The *when* parameter deals with the time when the adaptation occurs. We were first tempted to divide this criterion in static or dynamic, however this does not work, since these two terms are always related to something: a method which adapts each N received packets, for example, is static in terms of number of packets, and dynamic in time. So we decided to consider this criterion as static in the following terms: in time (e.g. in seconds), in number of RTTs, in number of packets, or other. Mathematically, this gives a function f(t) = k*t, f(RTT) = k*RTT, f(nbpkt) = k*nbpkt, or f(...) Note that some codecs may have constraints on the moment when the quality can change.

This parameter has a notable characteristic. Doing frequent adaptations increases indeed video quality received by the user, but has an adverse effect. Methods where the *when* parameter is small (e.g. up to a few seconds) might create undesirable oscillations in quality. Indeed, constantly adapting and changing video quality often leads to a negative perception of the video by the user. Methods to avoid such oscillations exist [40].

As shown in Table 2, several values are used in the literature for this parameter. Finding out the most appropriate value for *when* is still an open question.

Interval: There is no consensus on the value of this parameter. Some methods in the literature use an average on either the last RTT, or the whole interval. Others use the instant value, a fixed interval etc. Some articles, noted by N/A, have not presented it.

There are also methods, such as HTTP-based, which use several intervals, e.g. instant when using receiver buffer and whole interval when using network in seconds.

We are aware of the works on video bandwidth forecasting, in which the server introduces in the flow information about the following video frames, such as frame size, over an interval of one GOP for example, information which could subsequently be used by routers in the path to optimise the transfer [20]. It should be noted that this anticipation interval is not necessarily the same as the computing interval, which could itself be different than the interval where adaptation occurs, as explained in the previous section.

Properties: In the same table it can be seen that some methods use HTTP. This allows to bypass firewalls in various, currently very restrictive, networks (Internet, 3G

etc.) and provide a universal access service. Others use various transport protocols with congestion control (TCP, DCCP) or the classical feedback-featured RTP/RTCP. Others work with any transport protocol (TP) with congestion control (CC).

Additionally, several methods use multi-layer encoded videos (ML), which are currently seen much interest and provide good performance. Also, MPEG-21 standard support is provided by some methods too.

Given all the discussion above, we can wonder about what the *ideal method* is. Of course, it should be real-time, without beforehand data needed, so that it can be used in all cases, but this could sometimes be incompatible with other goals. In the following we propose a solution which we do not argue to be ideal, but could be valuable to study.

The parameters where the best value is still open question are *What*, *When* and *Interval*.

Who is a very important parameter. Methods based on server decision work in all cases, real-time and delay-tolerant transmission, since the server has information about video as soon as it is generated. On the other hand, methods based on client decision scale well, taking into account Internet resources (e.g. HTTP caches). A mixture of the two could potentially give the advantages of both solutions. For example, server sends small-size data information about video together with video data, and client takes the decision.

Properties parameter is very important too. Using HTTP solves connectivity issues at application level on currently-restrictive Internet. At transport level, TCP is connectivity-friendly too, but leads to non real-time behaviour. DCCP is also connectivity-friendly, since it is connection-oriented and has congestion control. Restrictive firewalls could be easily updated to allow DCCP. Therefore, a potential solution would be HTTP on DCCP for video transfers.

So our basic idea is that each HTTP transfer use TCP, as usually, except for video files, which use DCCP instead. Video files can be recognised using the extension of the file asked (for example .avi). Choosing DCCP instead of TCP can be done either on server (videos are always sent with DCCP) or on client (in the HTTP request header). Note that in DCCP data transfer is unreliable, but session establishment and teardown are reliable. We have already used DCCP (without HTTP) to stream video [39].

Summing up, a valuable solution we propose is to use HTTP on top of DCCP. The real-time goal could be obtained by making server send to client small-size data information about video together with video data, and let client take the decision about video adaptation.

7 Other surveys

In order to maximise the chances to find out related works, we have made a search using specific search terms and took into account all the articles returned. For the sake of precision, we describe here the method used.

An exhaustive search on *google scholar* was carried on March 2013 with the following search term: *allintitle: (multimedia OR video OR content) AND (adaptation OR adaptive OR adaptative) AND (survey OR classification OR taxonomy OR review OR comparison)*, which looks for articles containing in their title a word from each of the three OR groups[3]. Among all the papers found, we disregarded the ones not being a survey or which were completely irrelevant to our paper (for ex. papers about video *content* classification or which do not treat adaptation) or which were not written in English. Four papers remained, presented in the following.

Arsan [2] presents a review of bandwidth estimation tools for wired and wireless networks with respect to video streaming. The review is about estimating available bandwidth, which is one of the parameters used when doing video adaptation. This parameter corresponds to the *What* column of our analysis, when its value is *network bytes*, meaning that network information is used for adaptation. As such, it focuses on one particular part of our analysis.

Vandalore [49] is a very general survey on adaptive multimedia methods. It treats video compression standards such as MPEG and wavelet encoding, rate shaping[4], FEC, video data smoothing[5], video/audio synchronisation, operating system support (CPU time allocation, thread priority). Instead, we specifically detail parameters used in video adaptation.

Adzic et al. [1] consider the general case of Web page content adaptation (text, image, video etc.) to devices with specific constraints (small screen and bandwidth). They consider a full presentation, with video, audio, text, thumbnail etc. and the adaptation consists in cutting one of them (e.g. a proxy which removes all images from a Web page), modifying images so that they do not surpass screen size, adapting font size etc. The papers included in their survey do not adapt videos, but Web pages, contrary to our article. Hence, the two articles treat adaptation on different levels.

Pu [38] presents a survey on QoS mechanism meant for a distributed multimedia server, not for the transmission itself. He concentrates on middleware level, and considers various programming technologies, such as Java, CORBA and green threads to increase performance of multimedia server.

Sadaf [44] presents methods to choose a data rate (6, 9, ..., 54 Mbps) in 802.11 wireless networks function of the error rate of each data rate. The aim is to improve the video transmission from network point of view. It is specific to IEEE 802.11 networks: it takes into account channel availability, RTS/CTS and user mobility. This paper adapts network parameters, not data. Also, the scope of our article is broader (any network) and focuses on adaptation parameters.

[3] In practice, google scholar accepts only one group of ORs, so we divided the search term in 3*3=9 searches; for example one of them was *allintitle: multimedia adaptation survey OR classification OR taxonomy OR review*

[4] Rate shaping: adjust the rate of traffic generated based on the available bandwidth.

[5] Smoothing: in the context of VBR videos where different frames have different sizes in bytes, smoothing means sending parts of large frames later (or sooner), in the same time as smaller video frames, in order to use constant bandwidth.

8 Conclusions

This article has presented several methods found in the literature aiming to adapt video to network conditions during video streaming. A taxonomy of parameters taken into account by decision methods has been introduced. Methods presented have been discussed with respect to the taxonomy.

It turns out that many parameters can be used when taking the decision of video adaptation. In the research community, usually the sender does the adaptation. Major companies solutions currently use HTTP instead, and the adaptation is receiver-driven. Three parameters are commonly used for the decision: sender buffer, network or receiver buffer. Values can be measured either in bytes or in seconds of video.

After method analysis, a valuable and realistic method for video adaptation, taking into account advantages from several methods, has been suggested. It uses HTTP on top of DCCP with a small traffic from server to client, who takes adaptation decision.

We plan to analyse this ideal solution. A future work is also to analyse the "after decision" timeframe, dedicated to the actions to be done once the decision has been made: which video parameter to change, how and how much.

Author contributions

E. Dedu wrote introduction and other surveys sections. W. Ramadan worked on method presentation section. E. Dedu and W. Ramadan worked on context and motivations sections, and criteria description and discussion sections. J. Bourgeois provided sound comments on the whole article.

References

1. V. Adzic, H. Kalva, and B. Fuhrt. A survey of multimedia content adaptation for mobile devices. *Multimedia Tools and Applications*, 51(1):379–396, Jan. 2011.
2. T. Arsan. Review of bandwidth estimation tools and application to bandwidth adaptive video streaming. In *International Conference on High Capacity Optical Networks and Enabling Technologies (HONET)*, 9, pages 152–156, Istanbul, Turkey, Dec. 2012.
3. A. Balk, D. Maggiorini, M. Gerla, and M. Y. Sanadidi. Adaptive MPEG-4 video streaming with bandwidth estimation. In *International Workshop on Quality of Service in Multiservice IP Networks*, 2, pages 525–538, London, UK, February 2003. Springer-Verlag.
4. I. S. Burnett, F. Pereira, R. V. de Walle, and R. Koenen, editors. *The MPEG-21 Book*. Wiley, 2006.
5. J. M. Cubero, J. Gutiérrez, P. Pérez, et al. Providing 3D video services: The challenge from 2D to 3DTV quality of experience. *Bell Labs Technical Journal*, 16(4):115–134, Mar. 2012.
6. L. C. Daronco, V. Roesler, J. V. de Lima, and R. Balbinot. Quality analysis of scalable video coding on unstable transmissions. *Multimedia Tools and Applications*, pages 1–27, 2011. Available online.
7. L. De Cicco and S. Mascolo. An experimental investigation of the Akamai adaptive video streaming. In *6th international conference on HCI in work and learning, life and leisure: workgroup human-computer interaction and usability engineering*, USAB'10, pages 447–464, Berlin, 2010. Springer-Verlag.
8. L. De Cicco, S. Mascolo, and V. Palmisano. Feedback control for adaptive live video streaming. In *Second annual ACM conference on Multimedia systems*, pages 145–156, New York, NY, USA, 2011. ACM.

9. G. V. der Auwera, P. T. David, M. Reisslein, and L. J. Karam. Traffic and quality characterization of the H.264/AVC scalable video coding extension. *Advances in Multimedia*, 2008:1–27, 2008.
10. G. V. der Auwera and M. Reisslein. Implications of smoothing on statistical multiplexing of H.264/AVC and SVC video streams. *IEEE Transactions on Broadcasting*, 55(3):541–558, Sept. 2009.
11. Dynamic streaming in Flash Media Server 3.5. Available at `http://www.adobe.com/devnet/flashmediaserver/`, 2010. Adobe Systems Inc.
12. M. Eberhard, C. Timmerer, H. Hellwagner, and E. Quacchio. An interoperable delivery framework for scalable media resources. *Wireless Communication*, 16:58–63, October 2009.
13. W. Feng. On the efficacy of quality, frame rate, and buffer management for video streaming across best-effort networks. *Journal of High Speed Networks*, 11:199–214, 2002.
14. W. Feng and J. Rexford. Performance evaluation of smoothing algorithms for transmitting prerecorded variable-bit-rate video. *IEEE Transactions on Multimedia*, 1(3):302–312, Sept. 1999.
15. S. Floyd, M. Handley, J. Padhye, and J. Widmer. TCP Friendly Rate Control (TFRC): Protocol Specification. RFC 5348 (Proposed Standard), Sept. 2008.
16. M. Furini and D. Towsley. Real-time traffic transmission over the Internet. *IEEE Transactions on Multimedia*, 3(1):33–40, Mar. 2001.
17. B. Görkemli and A. M. Takalp. Adaptation strategies for MGS scalable video streaming. *Signal Processing: Image Communication*, 27(6):595–611, July 2012.
18. B. Gorkemli and A. M. Tekalp. Adaptation strategies for streaming SVC video. In *ICIP, The International Conference on Image Processing*, pages 2913–2916, Hong Kong, Septembre 2010.
19. E. Gürses, G. B. Akar, and N. Akar. A simple and effective mechanism for stored video streaming with TCP transport and server-side adaptive frame discard. *Computer Networks*, 48:489–501, July 2005.
20. R. J. Haddad, M. P. McGarry, and P. Seeling. Video bandwidth forecasting. *IEEE Communications Surveys & Tutorials*, PP(99):1–16, Apr. 2013.
21. N. Hu and P. Steenkiste. Evaluation and characterization of available bandwidth probing techniques. *IEEE Journal on Selected Areas in Communications*, 21(6):879–894, 2003.
22. A. Huszák and S. Imre. Selective retransmission of MPEG video streams over IP networks. In *International Symposium on Communication System Networks and Digital Signal Processing (CSNDSP)*, 5, pages 125–128, Patras, Greece, July 2006.
23. Information technology – dynamic adaptive streaming over HTTP (DASH) – part 1: Media presentation description and segment formats. ISO standard, Nov. 2011. ISO/IEC 23009-1:2012.
24. M. I. Kazantzidis. *Adaptive Multimedia in Wireless IP Networks*. PhD thesis, University of California, Los Angeles, USA, 2002.
25. I. Kofler, J. Seidl, C. Timmerer, H. Hellwagner, I. Djama, and T. Ahmed. Using MPEG-21 for cross-layer multimedia content adaptation. *Signal, Image and Video Processing*, 2(4):355–370, 2008.
26. P. A. Laplante and S. J. Ovaska. *Real-Time Systems Design and Analysis: Tools for the Practitioner*. Wiley, fourth edition, 2011.
27. S. Lee and K. Chung. MaVIS: Media-aware video streaming mechanism. In *IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services (MMNS)*, pages 74–84, Dublin, Ireland, October 2006.
28. S. Lee and K. Chung. Buffer-driven adaptive video streaming with TCP-friendliness. *Computer Communications*, 31:2621–2630, June 2008.
29. A. Lie and J. Klaue. Evalvid-RA: trace driven simulation of rate adaptive MPEG-4 VBR video. *Multimedia Systems*, 14(1):33–50, June 2008.
30. C. Liu, I. Bouazizi, and M. Gabbouj. Advanced rate adaption for unicast streaming of scalable video. In *IEEE International Conference on Communications*, pages 1–5, Cape Town, South Africa, May 2010.
31. C. Liu, I. Bouazizi, and M. Gabbouj. Rate adaptation for adaptive HTTP streaming. In *Second annual ACM conference on Multimedia systems*, pages 169–174, New York, NY, USA, 2011. ACM.
32. S. Mascolo. Congestion control in high-speed communication networks using the Smith principle. *Automatica*, 35(12):1921 – 1935, 1999.
33. D. T. Nguyen and J. Ostermann. Congestion control for scalable video streaming using the scalability extension of H.264/AVC. *IEEE Journal of Selected Topics in Signal Processing*, 1(2):246–253, 2007.
34. S. Oh, B. Kulapala, A. W. Richa, and M. Reisslein. Continuous-time collaborative prefetching of continuous media. *IEEE Transactions on Broadcasting*, 54(1):36–52, Mar. 2008.
35. R. Pantos and W. May. HTTP live streaming. IETF Draft, Mar. 2011. Apple Inc.
36. J. Postel. User Datagram Protocol. RFC 768 (Standard), Aug. 1980.

37. J. Postel. Internet Protocol. RFC 791 (Standard), Sept. 1981. Updated by RFC 1349.
38. J. Pu. A survey of the QoS adaptation mechanisms for distributed multimedia systems. Technical report, University of Victoria, Victoria, Canada, Jan. 2000.
39. W. Ramadan, E. Dedu, and J. Bourgeois. VAAL, video adaptation at application layer and experiments using DCCP. In *International Symposium on Wireless Personal Multimedia Communications (WPMC)*, 13, pages 1–5, Recife, Brazil, Oct. 2010. Springer.
40. W. Ramadan, E. Dedu, and J. Bourgeois. Avoiding quality oscillations during adaptive streaming of video. *International Journal of Digital Information and Wireless Communications (IJDIWC)*, 1(1):126–145, Nov. 2011.
41. W. Ramadan, E. Dedu, and J. Bourgeois. Oscillation-free video adaptation at application layer on server side and experiments using DCCP. *The Computer Journal*, 2013. Oxford University Press. To appear.
42. W. Ramadan, E. Dedu, D. Dhoutaut, and J. Bourgeois. RELD, RTT ECN Loss Differentiation to optimize the performance of transport protocols on wireless networks. *Telecommunications Systems*, 2011. Available online.
43. M. Roccetti, V. Ghini, G. Pau, P. Salomoni, and M. E. Bonfigli. Design and experimental evaluation of an adaptive playout delay control mechanism for packetized audio for use over the Internet. *Multimedia Tools and Applications*, 14(1):23–53, May 2001.
44. T. Sadaf, S. Zareen, F. Iqbal, G. A. Shah, and M. Y. Javed. Link adaptive multimedia encoding in wireless networks: A survey of theory and approaches. In *International Conference on Electronics and Information Engineering*, volume 2, pages V2–5–V2–10, Kyoto, Japan, Aug. 2010. IEEE.
45. T. Schierl and T. Wiegand. H.264/AVC rate adaptation for internet streaming. In *14th International Packet Video Workshop (PV)*, Irvine, CA, USA, December 2004.
46. H. Schwarz, D. Marpe, and T. Wiegand. Overview of the scalable video coding extension of the H.264/AVC standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1103–1120, Sept. 2007.
47. T. Stockhammer. Dynamic adaptive streaming over http — standards and design principles. In *ACM conference on multimedia systems*, 2, pages 134–144, Santa Clara, CA, USA, Feb. 2011.
48. Y.-C. Su, C.-S. Yang, and C.-W. Lee. Optimal FEC assignment for scalable video transmission over burst error channel with loss rate feedback. *Signal Processing: Image Communication*, 18(7):537–547, Aug. 2003.
49. B. Vandalore, W. chi Feng, R. Jain, and S. Fahmy. A survey of application layer techniques for adaptive streaming of multimedia. *Real-Time Imaging*, 7(3):221–235, June 2001.
50. N. Wakamiya, M. Murata, and H. Miyahara. TCP-friendly video transfer. In *Internet Quality and Performance and Control of Network Systems*, volume 4211, pages 25–35. SPIE, 2001.
51. M. Wien, R. Cazoulat, A. Graffunder, A. Hutter, and P. Amon. Real-Time system for adaptive video streaming based on SVC. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1227–1237, 2007.
52. M. Wien, H. Schwarz, and T. Oelbaum. Performance analysis of SVC. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1194–1203, Sept. 2007.
53. B. Xie and W. Zeng. Rate-distortion optimized dynamic bitstream switching for scalable video streaming. In *IEEE ICME, International Conference on Multimedia and Expo*, volume 2, pages 1327–1330, Taipei, Taiwan, 2004.
54. D. Ye, J. C. Barker, Z. Xiong, and W. Zhu. Wavelet-based VBR video traffic smoothing. *IEEE Transactions on Multimedia*, 6(4):611–623, Aug. 2004.
55. D. Ye, X. Wang, Z. Zhang, and Q. Wu. A buffer-driven approach to adaptively stream stored video over internet. In *International Conference on High Speed Networks and Multimedia Communications*, pages 81–85, Beijing, China, 2002.
56. A. Zambelli. IIS Smooth Streaming technical overview, Mar. 2009. Microsoft Corporation.