Adaptation strategies for MGS scalable video streaming

Burak Görkemli *, A. Murat Tekalp

College of Engineering, Koç University, 34450 Istanbul, Turkey

ARTICLE INFO

Article history:
Received 18 May 2011
Accepted 8 February 2012
Available online 18 February 2012

Keywords:
Adaptive streaming
Scalable video coding
Medium grain scalability (MGS)
Transport protocols
Adaptation strategy

ABSTRACT

An adaptive streaming framework consists of a video codec that can produce video encoded at a variety of rates, a transport protocol that supports an effective rate/congestion control mechanism, and an adaptation strategy in order to match the video source rate to the available network throughput. The main parameters of the adaptation strategy are encoder configuration, video extraction method, determination of video extraction rate, send rate control, retransmission of lost packets, decoder buffer status, and packetization method. This paper proposes optimal adaptation strategies, in terms of received video quality and used network resources, at the codec and network levels using a medium grain scalable (MGS) video codec and two transport protocols with built-in congestion control, TCP and DCCP. Key recommendations are presented to obtain the best results in adaptive video streaming using TCP or DCCP based on extensive experimental results over the Internet.

1. Introduction

Traditionally, video transport has been realized over dedicated, fixed bandwidth channels, such as terrestrial, satellite, or cable. Hence, video coding standards like MPEG-2 and MPEG-4 AVC encode video at a fixed target rate for a given resolution and application. With the advent of video over IP and WebTV, video transport must now be achieved over heterogeneous IP networks including a variety of fixed and wireless links. It is well-known that achieving the best video quality in this heterogeneous environment requires an adaptive streaming framework that can most efficiently adapt the video source rate to the available network throughput. Adaptive streaming can be implemented in both sender-controlled and receiver-controlled scenarios. In traditional sender-driven adaptive media streaming, a central server is in charge of estimating the available bandwidth and adapting the video rate for each client individually. The clients only aid the streaming server in bandwidth estimation, while the adaptation logic resides on the server. On the other hand, HTTP Live Streaming [1] describes a protocol for receiver-controlled streaming, where the adaptation logic is shifted to clients. The server is only responsible for servicing incoming client requests, while clients estimate the available network bandwidth and request appropriate chunks of media.

Fundamental blocks of an adaptive streaming framework, depicted in Fig. 1, are a codec that can output video at multiple rates, a transport protocol that employs effective rate/congestion control, and an adaptation engine, which can reside on the server or the client.

The video coding method has an important impact on the adaptation strategy to be employed. An overview of adaptation methods for different encoding options is given in [2]. If the video is encoded in multiple simulcast streams with varying bitrates, the streaming application can switch between the bit-streams according to network conditions, which can be initiated either by the server [3] or the client [1]. Alternatively, the video can be encoded...
in a single scalable bit-stream with multiple layers. Earlier approaches to scalable video coding included coarse-grain scalability (typically two-layers) and fine-grain scalability (FGS) which is byte-based quality scalability [4]. Scalable Video Coding (SVC) is the most recent related standard which features spatial, temporal and quality scalability, both on coarse-grain and medium-grain levels (MGS) [5]. Methods for spatial, temporal, SNR, priority-based, and ROI-based adaptation of an SVC bit stream are presented in [6]; while the spatial-temporal resolution of a scalable video is modified depending on the estimated bandwidth in [7]. Yet another video coding option for adaptive streaming is real-time encoding with effective rate control, which can be implemented with both non-scalable [8] and scalable [9] video coding.

Being the de facto transport protocol of the Internet, TCP is the first that comes to mind when transport protocols are considered. TCP is commonly used for streaming stored media with its built-in congestion control, reliable transmission and firewall friendliness. Popular video distribution sites, such as YouTube, Vimeo and Metacafe, use HTTP over TCP to stream video. It has been shown in [10] that using TCP for video streaming provides good performance when the available network rate is about twice the maximum video rate, with a few seconds pre-roll delay. In [11], authors aim to minimize the latency caused by large TCP send buffers by tuning the send buffer size to follow the TCP congestion window (CWND). However, TCP has some undesirable properties for video streaming in its design, such as lack of control on delay (especially for live video with strict end-to-end delay requirement) and rapidly changing transmission rate.

UDP is a transport protocol, which does not include TCP’s built-in congestion control and reliable, in order packet delivery, leaving their implementation to the application layer. Congestion control schemes that can be implemented with the UDP are listed in [12,13]. They can be classified into four groups: TCP’s Additive Increase Multiplicative Decrease (AIMD) based schemes, equation based schemes, the bandwidth estimation scheme employed in the TCP Westwood, and receiver buffer occupancy based schemes. The TCP Friendly Rate Control (TFRC) is a well-known equation based congestion control scheme designed for unicast flows, which uses the TCP throughput equation for calculating the allowed TCP friendly send rate [14].

The Datagram Congestion Control Protocol (DCCP) [15] is a more recent transport protocol, implementing bi-directional unicast connections of congestion-controlled, unreliable datagrams. DCCP features a choice of congestion control mechanisms, to be selected at startup, which are TFRC identified by Congestion Control Identifier 3 (CCID3) [16] and TCP-like Congestion Control identified by CCID2 [17]. TCP-like Congestion Control is similar to TCP’s AIMD mechanism, halving the congestion window in response to congestion. DCCP is designed for streaming media applications, which do not prefer TCP due to arbitrary long delays introduced by reliable in-order delivery and congestion control, and which do not wish to implement complex congestion control mechanisms needed with the UDP. The performance of streaming video over DCCP, UDP, and the Stream Control Transmission Protocol (SCTP) are compared in [18] and it is concluded that DCCP achieves better results than SCTP and UDP. Similar results are given in [19], where it is stated that using DCCP as alternative to UDP may lead to efficient throughput and lesser packet loss at the cost of increased jitter and delay. Moreover, it is shown in [20] that using DCCP for streaming in wireless networks yields better video quality than using UDP.

In adaptive video streaming, one of the main parameters to estimate is the available network rate, or the desired video send rate. In case DCCP CCID3 is used, the TFR rate calculated by DCCP can be utilized by the sender to estimate the available network rate, without requiring an additional estimation mechanism. Otherwise, the video send rate can be estimated using receiver buffer occupancy information to prevent any buffer underflow/overflow [12,21] or by combining the receiver buffer state with the bandwidth estimate [7]. A virtual network buffer between the sender and receiver is employed in [3] for video rate adaptation together with an end-to-end delay constraint; while the same virtual network buffer algorithm is used in [13] to implement source rate control and congestion control jointly. Packets may be sent depending on their rate distortion contributions [22,23]. A similar approach is presented in [24], where the expected distortion is minimized through optimal selection of the packets to be transmitted. Finally, Video Transport Protocol (VTP) is proposed in [25], to maximize the quality of real-time MPEG-4 video streams while simultaneously providing basic end-to-end congestion control. VTP behaves similar to AIMD in the sense that it performs additive increase but not multiplicative decrease: The protocol adjusts its transmission rate to the rate perceived by the receiver when there is congestion.

In this work, optimal adaptation strategies, in terms of received video quality and used network resources, for streaming SVC coded video over DCCP and TCP are proposed, expanding on and finalizing our previous works [26,27], supported by extensive test results over the Internet. The main contributions of this paper are:

- to determine the optimal adaptation strategy in the video coding domain; including determination of the optimal SVC MGS encoder configuration and extraction method.
- to determine the optimal adaptation strategy considering video coding and transport domains jointly; including the extraction and send rate control, packetization, and an adaptive Automatic Repeat reQuest (ARQ) scheme.
We note that while there have been many previous studies for adaptive video streaming based on MPEG-4 FGS coding, adaptation strategies for byte-scalable MPEG-4 FGS video are different than those for the currently used packet-scalable MPEG SVC video that are studied in this paper due to granularity of adaptation. MPEG-4 FGS allows arbitrary truncation according to the given rate budget, where any target rate within the video encoding range is precisely achieved. For MGS SVC, however, truncation is performed on packet-level, resulting in a much coarser rate adaptation than that of MPEG-4 FGS.

The paper is organized as follows: Section 2 discusses adaptation strategies related to SVC MGS encoder configuration and extraction methods. Section 3 describes the adaptation strategies related to video streaming system parameters. Section 4 presents extensive test results for the proposed adaptation strategies both on a local area network and over the Internet. Finally, Section 5 provides a discussion of the experimental results and presents key recommendations and conclusions.

2. Adaptation strategy—SVC encoder configuration and extraction methods

SVC provides quality scalability in two modes: coarse-grained scalability (CGS) and medium-grained scalability (MGS) [5]. A CGS layer cannot be partially retained or removed; hence, adaptation should be performed on complete layer basis and only a small number of different bit rates can be selected during rate adaptation. The MGS concept allows any enhancement layer Network Abstraction Layer (NAL) unit to be discarded from a quality scalable bit stream, providing packet-based scalability [28]. In MGS, it is possible to split zig-zag scanned transform coefficients into a number of fragments and form a separate sub-layer for each of the resulting fragments. This way, an MGS layer can be split into 2 to 16 sub-layers. It is also possible to put all coefficients into a single layer, without any fragmentation. In this work, the MGS scheme is employed to provide packet-based rate adaptation.

2.1. Encoder configuration

There are a number of MGS encoder configuration decisions that affect adaptive streaming performance. One of them is how to fragment the MGS layer into sub-layers. As an MGS layer is split into sub-layers, the number of available NAL unit combinations that can be selected in extraction increases, thus increasing the rate adaptation points. However, it should be noted that each sub-layer added introduces a cost in terms of rate-distortion. Hence, it is important to achieve a balance between adaptation and rate-distortion (R-D) performance.

Slice mode is another important parameter for configuring the encoder, enabling a video frame to be split into slices to minimize the effect of packet loss on received video quality. It is highly probable for some video packets to be lost or delayed during streaming, no matter what retransmission or forward error correction techniques are deployed. In such a scenario, dividing video frames into slices will limit the effect of packet loss to the corresponding slice, instead of affecting the entire frame. Again, there is a tradeoff between slice size and R-D performance.

Slicing can be done by limiting either the number of macroblocks or the number of bytes carried in a slice. When streaming over UDP or DCCP, it is preferable to use slicing in byte-limitation mode and keep the slice sizes less than 1500 bytes, which is the conventional MTU length. In this way, each slice is carried in a single transport packet. On the other hand, when TCP is used, slicing does not improve the received video quality, since the protocol does not preserve packet boundaries. Slicing can even decrease the video quality due to the overhead introduced in encoding. Hence, when streaming over TCP, it is advised to keep one slice per frame and disable slice mode.

In addition to limiting the effects of packet loss, slicing also increases the granularity of an MGS layer; because each layer is carried in multiple NAL units, rather than being carried in a much bigger single NAL unit. Rate adaptation can then be performed by selecting a set of slices belonging to a layer, in case it is not possible to extract that particular layer as a whole due to rate constraints. Since each slice data corresponds to a specific region in a frame, a quality enhancement gained by selecting a set of slices will only affect the corresponding regions, which may result in undesired quality variations over a frame.

2.2. Video extraction methods

In adaptive streaming of SVC video, it is a common practice to extract video on the basis of a Group of Pictures (GoP). That is, each GoP is extracted independently to satisfy a given rate constraint. All NAL units belonging to the base layer of a GoP must be extracted. There are various methods to select the NAL units to be extracted for enhancement layers. The simplest method, called flat-quality extraction, distributes the remaining rate budget among the enhancement layers of all frames in a GoP such that each frame has the same number of quality layers, if the budget allows. In case of using hierarchical B-pictures, the budget distribution will be in the order of temporal layers for each quality layer. The SVC reference software, called the Joint Scalable Video Model (JSVM), implements a version of this method, where all frames are extracted at the same quality layer.

Priority-based extraction is another extraction scheme having a better rate-distortion performance from flat-quality extraction for most of encoding configurations, which uses priority identifiers for each NAL unit that can be inserted as a post-encoding process [29]. The priority identifier, ranging from 0 to 63 with increasing priority, signals the contribution of the corresponding NAL unit to the video quality. The extraction method starts with selecting the NAL units with the highest priority identifier and continues in decreasing order, until the rate constraint is exceeded. It is evident that this method requires the priority information to be inserted in the video stream during encoding. There are also other rate-distortion based extraction methods, such as [30].
We propose an improved priority-based extraction strategy, called priority-based hierarchical extraction, which takes advantage of the hierarchical B-pictures structure together with the priority information. We start with the lowest temporal layer \( t_0 \) to select NAL units based on their priority identifiers and continue with the remaining temporal layers, \( t_1, t_2 \), and so on, as long as the rate budget allows. In case priority information is not present in the video sequence, our method reduces to the hierarchical extraction that is proposed in [28].

3. Adaptation strategy—streaming parameters

The block diagram of an adaptive video streaming system, which consists of a sender and a receiver module, is shown in Fig. 2. It can use DCCP (both CCID2 or CCID3) or TCP as the transport protocol. The sender estimates the available network rate, extracts a GoP of video using this target rate, and sends the extracted packets. The receiver inserts the arriving packets into a decoder buffer and then decodes the video for display. Main parameters of the streaming framework are given in Table 1. The rest of this section covers the details of video rate adaptation strategies related to these parameters.

3.1. Determination of the video extraction rate

When DCCP CCID3 is used as the transport protocol, the DCCP module at the sender periodically calculates the TFRC rate using receiver feedback and employs this rate to transmit the packets in its queue. This rate is also exposed to requesting applications. Hence, an adaptive streaming application can utilize the TFRC rate to discover the network dynamics and adjust video source rate accordingly, without any application-layer module to estimate the current bandwidth.

It is of interest to examine how to use the received TFRC rates in computing the rate of video extraction; that is, the relation between the send and the extraction rates. Since network condition is changing instantaneously, it may be

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r_{send}, R_{send} )</td>
<td>The instantaneous and the average rate, respectively, at which the sender application deposits video packets to the transport protocol.</td>
</tr>
<tr>
<td>( r_{resend}, R_{resend} )</td>
<td>The instantaneous and the average rate, respectively, at which the sender application deposits lost video packets to the DCCP. Packet loss occurs only when DCCP is used for transport; hence retransmission rates are defined for DCCP, not for TCP. In case of TCP, these rates are zero.</td>
</tr>
<tr>
<td>( r_{extract}, R_{extract} )</td>
<td>The instantaneous and the average rate, respectively, at which the video data sent is extracted.</td>
</tr>
<tr>
<td>( r_{tfrc}, R_{tfrc} )</td>
<td>The instantaneous and the average TFRC rate, respectively. ( r_{tfrc} ) is calculated and reported by the DCCP module. In case of TCP, these rates are zero.</td>
</tr>
<tr>
<td>( T_{average} )</td>
<td>The length of the averaging window, in milliseconds.</td>
</tr>
<tr>
<td>( T_{rc} )</td>
<td>Time at when the rate controller measures rates, in milliseconds.</td>
</tr>
<tr>
<td>( Q_{S_{min}} )</td>
<td>NAL unit count in the sender buffer, which triggers a new extraction.</td>
</tr>
<tr>
<td>( L_{play} )</td>
<td>Playout start threshold, in GoPs.</td>
</tr>
<tr>
<td>( L_{overflow} )</td>
<td>Decoder buffer overflow threshold, in GoPs.</td>
</tr>
</tbody>
</table>

Fig. 2. Block-diagram for a sender-driven adaptive video streaming system.
reasonable to average a series of TFRC rates over a time–window, for smoothing. The length of this time–window \(T_{\text{average}}\) is a parameter that affects the performance of streaming: The longer the window, the more stable is the extraction rate, but also less responsive to network dynamics. Thus, how received video quality varies by \(T_{\text{average}}\) is an interesting research issue.

TCP controls its transmission rate using window based congestion/flow control schemes but it does not expose any rate to applications in contrast to DCCP CCID3. The same also holds for DCCP CCID2, where TCP is mimicked. Therefore, we propose to use the packet deposit rate \(r_{\text{send}}\) averaged over a time–window with length \(T_{\text{average}}\) to adapt the quality of streamed video to the available bandwidth, unless DCCP CCID3 is used.

In our system, the rate controller module in sender requests the TFRC rate \(r_{\text{tfrc}}\) from DCCP every \(T_{\text{rc}}\) milliseconds, when DCCP CCID3 is used. This module also measures packet deposit rates \(r_{\text{send}}\) and the extraction rate of the packets deposited, \(r_{\text{extract}}\) every \(T_{\text{rc}}\) milliseconds. The module then averages all these rates over a moving time–window of length \(T_{\text{average}}\) and calculates the rate of the video to be extracted using Eqs. (1)–(8), as shown in Fig. 3.

\[
R_{\text{send}}(t_{i-1}) = \frac{1}{K} \sum_{i=1}^{K} r_{\text{send}}(t_{i-1})
\]  
\[
R_{\text{resend}}(t_{i-1}) = \left\{ \begin{array}{ll}
\frac{1}{K} \sum_{i=1}^{K} r_{\text{resend}}(t_{i-1}) & \text{if DCCP CCID2 or CCID3 used} \\
0 & \text{if TCP used}
\end{array} \right.
\]  
\[
R_{\text{tfrc}}(t_{i-1}) = \left\{ \begin{array}{ll}
\frac{1}{K} \sum_{i=1}^{K} r_{\text{tfrc}}(t_{i-1}) & \text{if DCCP CCID3 used} \\
0 & \text{if TCP or DCCP CCID2 used}
\end{array} \right.
\]  
\[
R_{\text{extract}}(t_{i-1}) = \frac{1}{K} \sum_{i=1}^{K} r_{\text{extract}}(t_{i-1})
\]  
\[
t_{i-1} - t_{i-1} \leq T_{\text{average}} \quad t_{i-1} - t_{i-1-K} > T_{\text{average}}
\]  
\[
t_{i} - t_{i-K} < T_{\text{rc}}
\]  
\[
R_{\text{diff}}(t_{i-1}) = R_{\text{send}}(t_{i-1}) - R_{\text{resend}}(t_{i-1}) - R_{\text{extract}}(t_{i-1})
\]  
\[
R_{\text{extract}}(j) = \left\{ \begin{array}{ll}
R_{\text{tfrc}}(t_{i-1}) + \min(R_{\text{diff}}(t_{i-1}), 0) & \text{if DCCP CCID3 used} \\
R_{\text{send}}(t_{i-1}) + \min(R_{\text{diff}}(t_{i-1}), 0) & \text{if TCP or DCCP CCID2 used}
\end{array} \right.
\]

The Eqs. in (1)–(4) average \(K\) instantaneous samples with equal weight. The number of samples to average, denoted by \(K\), is determined by the averaging-window length \(T_{\text{average}}\), as in Eq. (5). Moreover, Eq. (6) ensures that averaging starts with most recent samples. Then, the average rates are used to calculate the next extraction rate, as given by Eqs. (7) and (8).

Given an extraction rate \(R_{\text{extract}}\), the layer extractor module extracts a GoP of video satisfying this rate constraint, whose NAL units are inserted to the sender buffer to be kept until sent to the transport protocol queue. As the NAL units are fetched from the buffer, packetized and sent, the buffer starts to get empty. When the number of NAL units in the sender buffer decreases below \(QS_{\text{min}}\), a new extraction rate is computed and fed to the layer extractor for a new GoP of video. The value of \(QS_{\text{min}}\) is selected such that it is small enough for using the most recent bandwidth information in computing the extraction rate \(R_{\text{extract}}\). But, \(QS_{\text{min}}\) should also be large enough so that sending \(QS_{\text{min}}\) NAL units takes longer than the time taken by each extraction process; and therefore, the buffer gets filled with new extracted NAL units before getting drained.

### 3.2. Send rate control

The actual sending operation is implemented by depositing video packets to the transport protocol queue at a rate of \(r_{\text{send}}\). The transport protocol module at the sender transmits packets in its queue to its peer at the receiver at a rate determined by the protocol. In case of DCCP CCID3, this rate is the calculated TFRC rate. If the sender application inserts packets to the transport protocol queue faster than the rate at which the packets in the queue are processed, the queue may become full due to being limited. In this case, sender will wait for the queue to become available, meaning that the fast sender will be slowed down by the transport protocol.

The sender cannot directly control the rate at which the transport protocol sends packets in its queue to the receiver. More specifically, the sender cannot send packets faster than the transport protocol allows. However, it can slow down the transmission by dispatching packets to the transport protocol queue at a lower rate than the allowed rate. This may be meaningful especially when the available bandwidth is more than the highest extraction rate of the video. It may be considered that streaming video at the extraction rate rather than the higher

![Fig. 3. Calculation of the next extraction rate.](chart.png)
available bandwidth would result in lower loss, leaving more bandwidth to competing flows. In order to investigate the results of sending rate control, we have implemented three selectable modes of rate control in the streaming system. They are: (i) sending extracted video at the available bandwidth rate (send @ max. rate); (ii) sending video at the extraction rate (send @ ext. rate); (iii) sending video at the available bandwidth rate but not more than the maximum extractable video rate (send @ lim. max. rate).

Moreover, the sender may have to slow itself down to prevent any receiver buffer overflow. In our proposed system, the playout simulator module estimates the receiver side decoder buffer status depending on the packets transmitted as well as the decoder buffer feedback sent by the receiver. Then, using this estimate, the rate controller decreases packet dispatch rate in case of a decoder buffer overflow risk.

3.3. Retransmission of lost packets

Although video rate is adapted to the network, packet losses still occur with the DCCP. As the receiver detects missing packets with enough time for playout, it may request them from the sender using an ARQ scheme. We examine three different ARQ schemes, which are: (i) request all missing packets (arq all); (ii) request missing packets for base layer only (arq base); (iii) request all or base layer missing packets, depending on the decoder buffer occupancy (arq adaptive). In the first scheme (arq all), all of the missing packets are requested for retransmission, provided that their playout times have not expired. In the second scheme (arq base), only the base layer packets, which are mandatory for the error-free video decoding, are requested by the receiver. The third scheme makes this decision adaptively, based on the decoder buffer occupancy. If there is a decoder buffer overflow risk, all missing packets are requested from the sender, so as to give time to the decoder to process buffered GoPs before receiving new ones. Otherwise, only base layer packets are asked for retransmission. Hence, when the bottleneck bandwidth is more than the maximum extractable video rate, the remaining bandwidth is utilized for achieving the maximum received video quality. At other times, the available bandwidth is used for transmitting the mandatory data.

3.4. Number of GoPs extracted at a time

As default, we extract one GoP of video at a time. Increasing the number of GoPs extracted at a time to two, three or more at the same extraction rate introduces an extra averaging, which may smooth the rate differences between consecutive GoPs. However, it may cause the system to be less responsive to rapid changes in the network.

3.5. Decoder buffer size/status

The decoder buffer size is given in terms of GoP count, since the decoding is performed GoP by GoP in the system. The default buffer size is set as 20 GoPs, which corresponds to approximately 10 s of video, for a 30 fps video with a GoP size of 16 frames. We note that the size of the decoder buffer determines the pre-buffering period, since video playout starts as soon as half of the buffer is filled. Also, the decoder buffer overflow threshold $L_{overflow}$ that is used to prevent decoder buffer overflow, varies with the buffer size. The default buffer size of 20 GoPs presents a good compromise between the received video quality and the pre-buffering period, as shown in Section 4.2.5.

3.6. Packetization

When DCCP is used, the extracted video data should be packetized prior to being sent. Maximum size of a DCCP packet payload, which is a NAL unit, is given by the path Maximum Transmission Unit (MTU) minus the packet header introduced. For example, if the MTU is 1500 bytes, the maximum possible length of a NAL unit will be 1452 bytes, which is MTU (1500 bytes) minus RTP + DCCP + IP headers (12 + 16 + 20 bytes). In order to minimize the effect of packet loss on video quality, the video to be streamed is coded in slices so that each NAL unit can fit to a DCCP packet. Hence, each NAL unit is sent in a single DCCP packet, without being fragmented. Moreover, smaller NAL units are aggregated in decoding order to fit into a packet, for minimizing the packet header overhead. RTP packetization is done as per [31]. The same packetization scheme is also used for TCP for compatibility issues. In order to send the packetized NAL data over TCP, the packets are framed according to [32].

4. Results

The Soccer and City sequences at CIF resolution and Soccer at 4CIF resolution are used throughout the tests—Soccer at 4CIF is used only in the Internet tests. Each video is looped 10 times, so as to have a 2881 frame sequence lasting for about 96 s. The looped sequences are encoded using JSVM version 9.19.3, to have a base and an MGS layer, with the quantization parameter equal to (38, 28) for the Soccer video and (36, 26) for the City video. These quantization values present a good compromise between the encoding efficiency and the required range of video quality. The GoP size is set as 16 frames, and the first frame of each GoP is coded as key picture so that error drift is constrained to a single GoP. Moreover, each frame is partitioned into multiple slices using the byte-limited slice mode so that the maximum NAL unit length is less than the default MTU value of 1500 bytes, together with the RTP, DCCP and IP headers, guaranteeing none of the NAL units will be fragmented in the IP layer.

In order to find out how to fragment the MGS layer optimally, a wide set of MGS fragmentation configurations are compared in terms of extracted video quality, using three different extraction methods. Based on the results presented in Section 4.1, the MGS layer of videos are fragmented into two sub-layers using the MGS fragmentation configuration of (6,10), meaning that the first six transform coefficients form the first MGS sub-layer and the remaining ten coefficients go into the second sub-layer.
The maximum extraction rates and the corresponding PSNR values over the whole video are shown in Fig. 4. It can be noted from the figure that the rate and PSNR values vary much more for the Soccer sequence, compared to the City video. This is because the Soccer video contains more movement and scene change than the City video, which is more or less static. Additionally, the minimum (base layer only) and the maximum (base+MGS) extractable rates (without packetization) of the sequences are listed in Table 2, together with the corresponding PSNR values. The videos can be extracted at almost any rate between these minimum and maximum values, thanks to MGS fragmentation and multiple slices. It should be noted here that the values given in Table 2 are averaged over the whole sequence, while the ones given in Fig. 4 are averaged over GoPs in each sequence.

The tests are initially run over a controlled LAN, in which the Internet traffic is simulated using the network simulator ns2. The ns2 version 2.33 is used in emulation mode so that packets from sender to receiver pass through ns2, being subject to loss or delay. For cross traffic PackMime-HTTP [33] module is utilized, generating web traffic, with the rate parameter set to 1, 5, and 10 HTTP requests per second for low, moderate, and high cross traffic, respectively. Tests are done with varying parameters, at different bottleneck bandwidths ranging from 500 kbps to 1500 kbps with a step size of 100 kbps (without packetization) of the sequences are listed in Table 2, together with the corresponding PSNR values. The videos can be extracted at almost any rate between these minimum and maximum values, thanks to MGS fragmentation and multiple slices. It should be noted here that the values given in Table 2 are averaged over the whole sequence, while the ones given in Fig. 4 are averaged over GoPs in each sequence.

Following the LAN tests, new tests are performed on the Internet between Koc University and Argela Technologies, both located in Istanbul. The Internet speed between these two locations varies between 1 Mbps and 5 Mbps. For the Internet tests, 15 tests are run for each test set, and the results are averaged prior to presentation.

4.1. SVC and scalable video extraction methods

In order to find the optimal MGS fragmentation configuration and the extraction method, the Soccer and City sequences are encoded with various configurations and the resulting videos are extracted GoP-by-GoP with bandwidth constraints ranging from the rate of base layer to the MGS layer, in 50 kbps increments. During an extraction, the same rate constraint is used for all the GoPs of the video and the extraction is done so that none of the extracted GoPs exceed the given constraint. After each extraction, the rate is incremented in 50 kbps and the process is repeated, until the maximum extractable video rate is met. Moreover, each extracted video is decoded and compared with the original sequence, to calculate its PSNR. Fig. 5 gives the PSNR values of such extractions, averaged over a range of rate constraints for each video, which is from 200 kbps to 1200 kbps for the Soccer sequence and 200 kbps to 700 kbps for the City sequence, all in 50 kbps increments, with byte-limited slice mode enabled. It is apparent from the figure that the proposed priority-based hierarchical extraction performs better than the other methods in most of the MGS configurations; while the flat-quality extraction, used by the JSVM software by default, performs the worst, especially in configurations having many MGS sub-layers. The performance of flat-quality extraction improves as the MGS layer is fragmented into fewer sub-layers. It is interesting to observe that the priority-based extraction results in lower PSNR values, as the number of MGS sub-layers decreases, in contrast to the flat-quality extraction.

MGS fragmentation has negative impact on the PSNR performance, due to the fragmentation overhead. Fragmenting MGS into more than five sub-layers results in noticeable PSNR decrease, especially for the Soccer sequence, as depicted in Fig. 5. Choosing an MGS configuration with two sub-layers gives near-optimal results for all extraction schemes, provided that the first sub-layer contains fewer transform coefficients than the other. On the other hand, the City sequence hides the penalty of MGS fragmentation, performing better with more MGS fragments, compared to Soccer, except for the flat-quality extraction.

The performance of the City sequence can be better observed in Fig. 6, where the PSNR vs. bandwidth graphs for the slice mode disabled and enabled cases are compared to that of the Soccer sequence, generated using the priority-based hierarchical extraction. Different from Fig. 5, where the PSNR value of each extracted video is averaged over a range of bandwidth values to achieve a single average PSNR, Fig. 6 shows the PSNR of each extraction that corresponds to a given bandwidth. Here, the poor extraction performance of the City sequence for the non-fragmented MGS configuration with slice mode disabled.
can be clearly seen. This performance decrease is due to
the fact that the sizes of the key frame enhancement NAL
units are too big to be extracted at low rates, since each
layer is packetized in a single NAL unit. As these NAL units
are fragmented to MGS sub-layers having smaller NAL
units, the resulting PSNR values increase, because some
NAL units belonging to these sub-layers can be extracted
within the given bandwidth constraint. Hence, as the
number of MGS fragments in the configuration increases,
the extraction performance improves at low bandwidth
values. Similar performance gain can be achieved by
enabling the slice mode, since large NAL units are split
into smaller ones whose sizes do not exceed the given byte
limit. Enabling slice mode for the City video increases the
PSNR values at low rates, compared to the non-sliced case,
except for the \(1,1,1,1,1,1,1,1,1,1,1,1,1,1,2\) configuration
where most NAL units are already under the given byte
limit without using slice mode.

The effect of enabling slice mode can also be observed
in Fig. 7, where the average PSNR values for the MGS
configurations with fewer fragments are increased for the
City video. On the other hand, the overhead of byte-
limited slice mode, however small, is seen for the Soccer
video. This overhead will increase as the byte limit, which
is 1400 bytes, is lowered.

Based on the results presented, the videos to be used in
streaming tests are encoded with the MGS fragmentation
configuration of \(6,10\) and byte-limited slice mode enabled.
While there are other fragmentation configurations showing comparable performance, especially the ones with more fragments, differences between them are minimal, and the configuration of \((6,10)\) results in just two fragments. Moreover, for video rate adaptation, the priority-based hierarchical extraction method is employed.

### 4.2. Determination of streaming schemes and framework parameters

This section covers the tests performed using various streaming schemes and framework parameters, using the Soccer and City videos that are extracted with the priority-based hierarchical extraction scheme depending on the available bandwidth.

#### 4.2.1. Determination of the video extraction rate

Tests have been performed with two different averaging window lengths \((1\text{ and }5\text{ s})\) using various transport protocols and their performances are compared with the results found without any averaging. Tests using DCCP CCID3 show that averaging the calculated TFRC rate over a window improves the quality of the received video. Increasing the averaging window length from 0 (no averaging) to 1 s enhances the video quality by approximately 0.3 dB for the Soccer video and 0.9 dB for the City video at maximum, as given in Fig. 8. Further increasing the window length to 5 s does not result in a significant difference, except for the high traffic case at 500 kbps when the City video is used, where extending the averaging window length to 5 s improves the video quality by 0.7 dB by smoothing out TFRC rate oscillations.

Averaging improves the received video quality much more than CCID3, when DCCP CCID2 is used, except for the highest congestion case (high traffic at 500 kbps). Fig. 9 shows that the gain earned by performing averaging over a 1 s window is about 3 dB for the Soccer video and 2.5 dB for the City video at maximum, when compared with the no-averaging results. This significant difference is due to the fact that the packet sending rates are averaged and used for extraction in CCID2, as opposed to the TFRC rates in CCID3. Since the video packets are dispatched to the network in a discrete approach,
measuring the sending rate instantaneously results in a noisy rate, which decreases the video quality when used for video extraction. High traffic case at 500 kbps is an exception to this, where only base layer video is sent because averaging is not applied and that is why the target extraction rate is mostly zero. This tells us that sending video at the base layer without any adaptation results in the best performance when the network is highly congested.

Increasing the averaging window length from 1 s to 5 s decreases the received video quality at low bandwidths: the difference is as much as 0.7 dB for the Soccer video and 0.5 dB for the City sequence at 500 kbps low traffic scenario. The quality difference between these averaging lengths diminishes as the bottleneck bandwidth increase or more traffic is introduced into the network.

Increasing the averaging window length from 1 s to 5 s decreases the received video quality at low bandwidths: the difference is as much as 0.7 dB for the Soccer video and 0.5 dB for the City sequence at 500 kbps low traffic scenario. The quality difference between these averaging lengths diminishes as the bottleneck bandwidth increase or more traffic is introduced into the network.

Packet send rate is used for calculating the video extraction rate also when TCP Reno is the transport protocol. Applying averaging over the packet send rate improves the received video quality significantly, as can be noted in Fig. 10. The quality difference between no-averaging and 1 s averaging is as much as 14 dB for the Soccer sequence and 10 dB for the City when the bottleneck bandwidth is low. Here, not averaging the packet send rate does not produce higher quality videos when the network is highly congested, as experienced with DCCP CCID2, because TCP sends packets in a more bursty scheme. As the bandwidth increases, the quality difference between the no-averaging and 1 s averaging scenarios drops to approximately 2 dB. Extending the averaging window length to 5 s improves the results slightly at most cases, whereas there are also times when unnoticeable decreases are experienced.

Results given in the figures show that the best performing averaging window length values are 5 s for the DCCP CCID3 and TCP Reno, while it is 1 s for the DCCP CCID2. These are the default values used for the averaging window length throughout the tests, unless otherwise stated.

4.2.2. Send rate control

Tests have been performed under varying bottleneck bandwidth and cross traffic, using the three rate control schemes: (i) send @ max. rate; (ii) send @ ext. rate; (iii) send @ lim. max. rate. Results given in Figs. 11 and 12 reveal that sending video at the maximum or limited maximum rate produces videos with similar quality, whereas sending at the extraction rate generates lower quality videos, especially when there is cross traffic. Additionally, Fig. 11 shows that limiting the video transmission rate (send @ lim. max. rate) results in less packet loss and therefore less retransmission traffic when the
available bottleneck bandwidth is more than the maximum extractable video rate.

When DCCP CCID3 is used with no cross traffic, the three rate control schemes show similar performance up to the bottleneck bandwidth of 800 kbps for the Soccer video, as seen in Fig. 11. At this rate, sending video at the extraction rate starts performing worse than the other two schemes: The PSNR difference is about 0.7 dB at 800 kbps and 0.9 dB at 900 kbps. The reason of this quality decrease can be observed Fig. 13, where the TFRC rate calculated by DCCP varies between 400 kbps and 1500 kbps when sending rate is kept at the video extraction rate; compared to the variation between 700 kbps and 900 kbps when video is sent at the limited maximum available rate. Keeping the transmission rate at the video extraction rate results in TFRC rate oscillating. This is due to the nature of TFRC rate calculation: Packet loss is used to estimate the available bandwidth, along with RTT, and when a source transmits data at a rate lower than the calculated TFRC rate, less number of packets gets lost and DCCP overshoots the bandwidth estimate. Then the sender node in our streaming framework uses this overshot estimate to extract new packets, many of which in turn will be lost. Hence this time DCCP will lower the TFRC rate, resulting in oscillations.

The oscillating behavior of TFRC starts to be experienced at bottleneck bandwidth value of 800 kbps for the Soccer video. Before that, the video is extracted at the bottleneck rate; hence there is no difference between sending at the extraction rate or the available rate. At 800 kbps, some parts of the video will be extracted below the available bandwidth, because their maximum extraction rates will be below 800 kbps, as shown in Fig. 4. The same holds for 900 kbps, 1000 kbps and 1100 kbps, where there is difference between the extraction rate and the available bandwidth. However, after 1200 kbps, although the extraction rate differs from the bottleneck bandwidth, the three sending rate control schemes start to give similar results, because the bandwidth is over-provisioned and the oscillating behavior of TFRC is compensated.

City video results show that the performance of sending rate control schemes differ based on content. When there is no cross traffic, the three schemes perform similarly in terms of received video quality for the City sequence, as seen in Fig. 11. Moreover, starting with the bottleneck bandwidth of 900 kbps, which is approximately the maximum extraction rate for the City video, sending at the extraction rate results in less retransmission traffic than the other rate control schemes.
After 1000 kbps, sending at the extraction rate results in no packet loss at all, while the other schemes, especially sending at the maximum available rate causes some packets to be dropped. However, as cross traffic is introduced into the network, sending at the extraction rate starts to perform worse than the other two schemes, similar to what is observed with the Soccer video.

Tests performed with DCCP CCID2 and TCP Reno show that controlling the transmission rate results in poorer performance, as given by Figs. 14 and 15. This is due to the fact that the extraction rate is calculated depending on the transmission rate, when DCCP CCID2 and TCP Reno are used. Sending video at the extraction rate causes a bursty transmission pattern, decreasing the calculated extraction rate and hence the extracted video quality.

4.2.3. Retransmission of lost packets

Performances of the existing ARQ schemes using DCCP CCID3 are shown in Figs. 16 and 17, without any cross traffic and under high cross traffic, respectively. The results reveal that resending all missing packets does not produce a noticeable increase in the received video quality when the bandwidth is scarce. Under cross traffic, resending all packets even causes minor quality decrease, despite the increased retransmission traffic. In case the available network rate is high, however, the highest possible video quality cannot be achieved by requesting only base layer missing packets, because of the lost enhancement layer packets. At this point, the proposed adaptive ARQ scheme aims to scale the retransmission traffic, requesting only base layer packets when the network is low and requesting all missing packets when the network allows. In this way, the unnecessary retransmission traffic can be avoided by 40% at best and the maximum video quality can be reached.

Existing ARQ schemes perform more or less the same in terms of received video quality, when DCCP CCID2 is utilized, as given in Fig. 18. Inability to reach to the maximum possible video quality, experienced with CCID3, is not seen with CCID2, due to the difference
between missing packet ratios, especially at high bandwidth. When CCID3 is used and only base layer missing packets are retransmitted, the missing packet ratio can be as high as 4%. However, this value is about 1% for DCCP CCID2 at high network rates, which does not result in a noticeable difference in video quality. This difference in missing packet ratios is due to CCID2’s AIMD behavior, which acts faster to changes in network conditions than CCID3’s TFRC scheme.

4.2.4. Extraction GoP size

Fig. 19 gives the received video quality values for different extraction GoP sizes: one GoP, two GoPs and five GoPs. It is apparent from the figure that one GoP and
two GoPs perform similar, while minimal quality decrease is experienced as the video extraction size is increased to five GoPs (approx. 2.7 s), due to the decreased adaptability. Hence, extraction size of one GoP is a reasonable value for adaptation to the available network rate. Similar results are achieved with DCCP CCID2 and TCP Reno, which are not shown here.

4.2.5. Decoder buffer size

Tests have been done with three different decoder buffer sizes: 40 GoPs, 20 GoPs, and 10 GoPs. Table 3 gives the buffer sizes and the corresponding $L_{\text{overflow}}$ and $L_{\text{play}}$ threshold values, in terms of GoPs and seconds.

$$\begin{array}{|c|c|c|c|c|c|}
\hline
\text{GoP} & \text{Buffer size} & \text{GoP} & \text{GoP} & \text{GoP} & \text{seconds} \\
\hline
1 & 40 & 2 & 20 & 2 & 10.7 & 10 & 5.3 \\
\hline
2 & 30 & 16 & 15 & 9 & 7 & 3.7 \\
3 & 20 & 10.7 & 10 & 5.3 & 5 & 2.7 \\
\hline
\end{array}$$

4.2.6. Comparison of protocols

This section compares the performances of DCCP CCID3, DCCP CCID2, and TCP Reno under varying cross traffic. Fig. 21 gives the PSNR values of the received video for each protocol used, from which the TCP's retransmission penalty can be clearly seen under heavy congestion. On the other hand, the protocols show similar performance when there is plenty of available network capacity. Both CCIDs of DCCP behave alike under high cross traffic, with CCID2 performing slightly better at the highest congestion. At other times, CCID3 shows a marginally better performance compared to CCID2, about 0.3 dB at maximum.

Fig. 22 shows pre-buffering periods for the tests, whose PSNR values are given in Fig. 21. The tests are done with a decoder buffer size of 20 GoPs, that is, the pre-buffering lasts until 10 GoPs are received. It is apparent from the figure that the shortest pre-buffering period is achieved with the DCCP CCID2, which is about 2.6 s at maximum.

4.3. Internet tests

Streaming tests are performed using DCCP CCID3 and TCP Reno, between Koc University and Argela Technologies, which are connected with a link of 5 Mbps approximately. DCCP tests are limited to CCID3, that is, DCCP CCID2 is not used in the Internet tests, since both CCIDs performed similar in the LAN tests.
Fig. 23 shows the quality of the received videos averaged over each test set, which comprises twenty tests run sequentially in a close time-frame, using DCCP CCID3 and TCP Reno in alternating fashion. That is, a DCCP CCID3 test is followed by a TCP Reno test, which is followed by CCID3, continuing until twenty tests are run, making ten for DCCP and TCP. The results of these ten tests are then averaged for each protocol, prior to being presented. Tests done without competing traffic between end-hosts reveal that DCCP and TCP perform similar when there is enough network capacity. Here, it should be noted that there still can be uncontrollable short-lived cross traffic between

<table>
<thead>
<tr>
<th>Decoder Buffer Sizes for DCCP CCID3 with High Traffic - Soccer Video</th>
<th>Decoder Buffer Sizes for DCCP CCID3 with High Traffic - City Video</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR (dB)</td>
<td>PSNR (dB)</td>
</tr>
<tr>
<td>40 GoPs</td>
<td>40 GoPs</td>
</tr>
<tr>
<td>20 GoPs</td>
<td>20 GoPs</td>
</tr>
<tr>
<td>10 GoPs</td>
<td>10 GoPs</td>
</tr>
<tr>
<td>400 500 600 700 800 900 1000 1100 1200 1300 1400 1500 1600 400 500 600 700 800 900 1000 1100 1200 1300 1400 1500 1600</td>
<td></td>
</tr>
<tr>
<td>Pre-buff. Period @ low traffic (sec)</td>
<td>Pre-buff. Period @ moderate traffic (sec)</td>
</tr>
<tr>
<td>0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80 82 84 86 88 90 92 94 96 98 100</td>
<td></td>
</tr>
<tr>
<td>Pre-buff. Period @ high traffic (sec)</td>
<td>Pre-buff. Period @ moderate traffic (sec)</td>
</tr>
<tr>
<td>0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80 82 84 86 88 90 92 94 96 98 100</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 20. Results for different decoder buffer sizes under high cross traffic with DCCP CCID3.

Fig. 21. PSNR values for DCCP CCID3, DCCP CCID2 and TCP Reno under varying cross traffic.

Fig. 22. Pre-buffering period in seconds for DCCP CCID3, DCCP CCID2 and TCP Reno under varying cross traffic, when the decoder buffer size is 20 GoPs and the playout start threshold is 10 GoPs.
the sender–receiver hosts, since the tests are done in the open Internet. When competing traffic is started alongside with the video stream, which is basically an FTP flow, DCCP performs better than TCP, as much as 1.6 dB at maximum in terms of received video quality. Both of these results—with and without competing traffic—agree with the controlled LAN tests. Moreover, it is apparent in Fig. 24 that TCP Reno leaves more bandwidth to the competing flow than DCCP CCID3, with DCCP still being fair to the competing flow. The average video-to-competing-flow ratio is 1.3 for DCCP, whereas it is 0.8 for TCP.

5. Discussion and conclusions

It is well-known that an adaptive streaming framework is needed to obtain the best performance (in terms of video quality) for video transport over heterogeneous IP networks. However, interactions between the video coding parameters, transport protocols, and the adaptation engine must be carefully analyzed. We can make the following observations and recommendations after analyzing the results of extensive tests presented in Section 4:

- We observe in Fig. 5 that MGS configurations with two sub-layers give near-optimal R-D results for all extraction schemes, provided that the first sub-layer contains fewer transform coefficients than the other. Hence, we recommend the MGS fragmentation configuration set equal to (6,10), meaning that the first six transform coefficients form the first MGS sub-layer and the remaining ten coefficients go into the second sub-layer.
- We observe in Fig. 5 that the proposed priority-based hierarchical extraction performs better than the other methods for most MGS configurations; while the flat-quality extraction performs the worst, especially in configurations having many MGS sub-layers.
- Enabling byte-limited slice mode decreases the average PSNR for the Soccer video, as seen in Fig. 7, due to the reduced encoding performance. But for the City video, whose key frame enhancement NAL units are relatively big, enabling slice mode increases the average PSNR, as given in Figs. 6 and 7, because the big NAL units are split into smaller ones, a set of which can be selected during extraction at low rates instead of selecting none when not split.
- Comparing the transport protocols, in terms of received video quality and pre-buffering period, it is evident that TCP performs very poorly under heavy congestion, which can be observed in Figs. 21 and 22, due to its retransmission policy. Otherwise, when there is plenty of available network capacity, the protocols show similar performance, in terms of received video quality. The tests also reveal that the quickest playout start can be achieved with the DCCP CCID2 and the longest pre-buffering is observed with TCP Reno, as shown in Fig. 22.
- For rate estimation, it has been observed in Figs. 9 and 10 that averaging improves the video quality considerably when DCCP CCID2 or TCP is utilized.
- For the send rate control, we recommend the following strategy: (i) If the network rate is higher than the maximum video rate, then send at the maximum video rate until the buffer fills up to $L_{\text{overflow}}$, and then reduce the sending rate to the current extraction rate. (ii) If the network rate is lower than the maximum video rate, then stop sending and wait for a new extraction rate to be selected.
rate, then send at the available network rate for all transport protocols, provided that the receiver buffer does not overflow. Oscillations have been observed in the TFRC rate in Fig. 13, when video is sent at the extraction rate using the DCCP CCID3.

- When dealing with packet losses, retransmitting only base layer packets is sufficient, except for the case when the network is over-provisioned. A better approach is retransmitting base or all lost packets, depending on the available network rate, as observed in Figs. 16 and 17. Hence, we propose an adaptive ARQ scheme, which aims to scale the retxmission traffic using the decoder buffer occupancy information.

In conclusion, we recommend the following optimal adaptation strategy: Assuming SVC coding with the recommended MGS configuration of (6,10) and byte-limited slice mode enabled (see Figs. 5–7); use DCCP CCID3 as the transport protocol (see Figs. 21–23); extract video using the proposed priority-based hierarchical extraction scheme, which aims to scale the retxmission traffic using the decoder buffer occupancy information (see Figs. 16 and 17).

References