Motion-based Adaptive Streaming in WebRTC using Spatio-Temporal Scalable VP9 Video Coding

Gonca Bakar, R. Arda Kirmizioglu, and A. Murat Tekalp
Dept. of Electrical and Electronics Engineering, Koc University
34450 Sariyer, Istanbul, Turkey
Email: gbakar15@ku.edu.tr, rkirmizioglu@ku.edu.tr, mtekalp@ku.edu.tr

***Abstract*—WebRTC has become a popular platform for real-time communications over the best-effort Internet. It employs the Google Congestion Control algorithm to obtain an estimate of the state of the network. The default configuration employs single-layer CBR video encoding given the available network rate with rate control achieved by varying the quantization parameter and video frame rate. Recently, some open-source WebRTC platforms provided support for VP9 encoding with spatial scalable encoding option. The main contribution of this paper is to incorporate motion-based spatial resolution adaptation for adaptive streaming rate control and evaluate the use of single-layer (non-scalable) VP9 encoding vs. motion-based mixed spatio-temporal scalable VP9 encoding in point-to-point RTC between two parties in the presence of network congestion. Our results show that, during intervals of high motion activity, spatial resolution reduction with sufficiently high frame rates and reasonable quantization parameter values yield more pleasing video quality compared to the standard rate control scheme employed in open-source WebRTC implementations, which uses only quantization parameter and frame rate for rate control.**

I. Introduction

Video telephony or videoconferencing over the open Internet or Web has become an essential means of real-time communications (RTC). There are a number of highly popular proprietary solutions available. A comparative study of some of these solutions can be found in [1]. WebRTC is a free, open project that aims at standardizing an inter-operable and efficient framework for Web-based RTC via simple application programming interfaces (API) using Real-Time Protocol (RTP) over User Datagram Protocol (UDP) for push-based video transfer over point-to-point connections.

Video conferencing applications over the best-effort Internet have to cope with user device and network access heterogeneities, and dynamic bandwidth variations. Congestion occurs when the amount of data being sent over a network link is more than the capacity of the link, which results in queuing delays, packet loss, and delay jitter. While seconds of buffering delay is tolerable in uni-directional video streaming, in interactive RTC the maximum tolerable delay is limited to a couple of hundreds of milli-seconds. Thus, RTC applications must employ powerful network estimation, congestion control, and video rate adaptation schemes. A comparison of receive-side congestion control algorithms for WebRTC has been reported in [2]. The Google Congestion Control algorithm that is employed in the WebRTC platform has been described and analyzed in [3].

Regarding the choice of video encoder, WebRTC "browsers" and (non-browser) "devices" are required to implement the VP8 video codec as described in RFC6386 [4] and H.264 Constrained Baseline as described in [5]. Recently support for VP9 encoding/decoding [6] and spatial scalable coding option with VP9 have been added in most platforms. This allows WebRTC users to have a choice between non-scalable vs. scalable coding options considering their associated tradeoffs. For coding video at full resolution (sum of all layers), scalable coding introduces a source bitrate overhead as compared to non-scalable encoding. This overhead is limited to 30% (in spatial scalability mode). When resolution is same, overhead is close to zero. However, when we transmit video over a congested network we have to account for packet losses and delay jitter. A non-scalable codec can handle those problems with FEC or sending I frames more frequently, which typically nullifies the bitrate advantage of non-scalable coding.

Scalable video coding in multi-party videoconferencing, where clients have heteregeneous devices and/or network connections, has been proposed and implemented for several years [8]. An MCU transcoding each stream to multiple resolutions and bitrates may actually require the lowest source bitrate; however, this comes at the cost of high computational power and delay. Furthermore, in multi-party RTC, the bitrate overhead due to spatial scalable encoding is only from the sender to the server (router), and not from the server to the receiver. Hence, the advantages of scalable coding in multi-party RTC is clear. This paper advocates that scalable video coding offers video quality advantages even for two-party point-to-point RTC in the presence of network congestion.

In this paper, we provide an evaluation of video quality for non-scalable vs. scalable coding with CBR and VBR encoding options at the same bitrate in the case of two-party point-to-point WebRTC conferencing in the presence of network congestion. Multi-party videoconferencing and associated design issues and problems are not addressed in this paper. Our results show that mixed spatio-temporal scalable coding together with our proposed motion-adaptive layer selection algorithm can significantly improve video quality under network congestion compared to single-layer rate control by only quality factor (quantization parameter) and/or temporal frame rate reduction.

II. Related Works

This section first provides an overview of the WebRTC iniative, and then discusses prior art in scalable VP9 video coding and rate control for mixed spatio-temporal resolution video encoding.

A. Overview of WebRTC

WebRTC is an open source technology that brings RTC capabilities to web browsers, non-browser applications, and mobile devices via APIs for fast and easy application development. RTC can be in the form of peer-to-peer (P2P) or multi-party audio/video conferencing. In the P2P mode, a signaling server is needed for only establishing connection between two users using Session Description Protocol (SDP) messages and create media paths. The source code is available from [9].

WebRTC provides media capture, audio/video stream and data sharing APIs for developers. Major components of WebRTC include: (i) getUserMedia, which allows a web browser to access the camera and microphone and to capture media, (ii) RTCPeerConnection, which sets up audio/video calls, and (iii) RTCDataChannel, which allows browsers to share data. WebRTC uses RTP network protocol implemented over UDP transport layer protocol. RTP is coupled with RTCP (Real-time Transport Control Protocol) to monitor and estimate available bandwidth between end systems [15]. WebRTC applications are required to support VP8 and H.264 video encoding. Some platforms also provide support for VP9 video coding with its spatial and temporal scalable coding extension.

WebRTC network packets can traverse entire network infrastructure including NATs (Network Address Translation) and firewalls. NATs map a group of private IP addresses to a single public IP address in a device such as a router or firewall. The main reason for this is that there are 2^{32} possible IPv4 addresses, which are about to be exhausted. STUN (Session Traversal Utilities for NAT) servers help end-points find each other in the presence of NATs. Firewalls used for security concerns that might drop specific flows or allow only specific flows. TURN (Traversal Using Relays around NAT) servers are used as relays in the presence of firewalls or NATs in order to establish a connection.

B. Scalable Coding in VP9

VP9 is an open and royalty-free video compression standard developed by the WebM project sponsored by Google. VP9 has been shown to outperform VP8 and H.264 in terms of rate-distortion performance at the expense of higher computational complexity. Yet, it is possible to perform real-time VP9 encoding/decoding at 30 Hz on a standard laptop for standard definition (SD) video using libvpx codec implementation. In addition to its better compression efficiency, VP9 also offers support for temporal and spatial scalable video coding.

A scalable video encoder produces multiple encoded bitstream layers, which depend on each other, forming a hierarchy. A specific layer, together with the layers it depends on, determines a particular spatial and temporal resolution. The layer that does not depend on any other layer determines the lowest spatio-temporal resolution and is called the base layer. Each additional layer improves spatial and/or temporal resolution of the video. A mixed (spatio-temporal) scalable encoded video with three spatial and three temporal layers and the dependency structure between the layers are depicted in Figure 1. In the figure S0 (red), S1 (purple) and S2 (pink) denote the spatial layers, while T0 (yellow), T1 (green) and T2 (blue) denote the temporal layers. The arrows show the dependencies between the layers, where white arrows show spatial dependencies and black arrows show temporal dependencies.

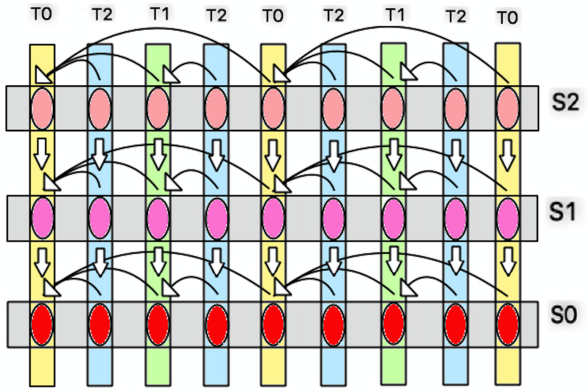

Fig. 1. Mixed spatio-temporal scalable coding with three spatial and three temporal layers and the dependency between the layers.

VP9 manages the spatial scalability structure by using super frames. A super frame consists of one or more layer frames, encoding different spatial layers. Within a super frame, a layer frame, which is not from the base layer, can depend on a layer frame of the same super frame with a lower spatial layer.

Two types of payload formats for a scalable VP9 stream are possible: flexible mode and non-flexible mode. In the flexible mode, it is possible to change layer hierarchies and patterns dynamically. In the non-flexible mode, the dependency structure and hierarchies within a group of frames (GOF) are pre-specified as part of the scalability structure (SS) data. SS data is sent with key frames once for each GOF and is also used to parse the resolution of each spatial layer. In this mode, each packet must have an index to refer to the spatial layer of it.

It is possible to achieve a coarse level source bitrate control, in order to match the source video bitrate to the estimated network bitrate, for adaptive video streaming by discarding one or more spatial and/or temporal layers per frame. Further finer bitrate control can be achieved by adjusting the quantization parameter value (quality level adaptation).

C. Mixed Spatio-Temporal Resolution Rate Control

In adaptive streaming, rate control at the encoder aims at adapting the video source bitrate to the estimated network bitrate. While rate control is a well studied problem in single resolution video coding, only few works exist for spatial-resolution adaptive (spatial scalable) video coding. On-the-fly rate-adaptation for low-delay scalable video coding has been proposed for the case of P2P push streaming over multi-cast trees [10]. Previous work has shown that spatial

resolution adaptation can improve rate-distortion performance and provide more consistent video quality over time at low bitrates [11], [12]. More recently, Hosking *et al.* [13] showed that rate control by spatial resampling can provide a higher and more consistent level of video quality at low bitrates in the case of intra-only HEVC coding. Motivated by these observations, this paper incorporates rate control by motion-based spatial resolution adaptation into the WebRTC platform and shows that more consistent and pleasing video quality can be achieved at low bitrates (in the presence of congestion).

III. The Proposed System

This section first describes the proposed system architecture for motion-based mixed spatio-temporal resolution rate control in a P2P WebRTC session, and then provides the details of the operation of the proposed motion activity detection and layer selection manager modules.

A. System Architecture

The bitrate of the mixed scalable coded source video must dynamically adapted to the changing network bitrate. As the network bitrate degrades, the source bitrate must be reduced accordingly to avoid delay jitter and packet losses. Alternatively, when the network bitrate improves, the source encoding bitrate should be increased up to the maximum encoding rate.

The reduction of video bitrate can be achieved by decreasing the frame rate, spatial resolution, or quality (increasing quantization parameter). The frame rate must be related to the amount of motion activity. When the video has high motion activity, reducing the frame rate by discarding temporal layers causes motion jitter which affects user experience negatively. It is well-known high spatial frequencies are not visible to human eye when the motion activity high due to the spatio-temporal frequency response of the human eye. Hence, we can reduce the spatial resolution of the video by discarding a spatial layer. On the other hand, the spatial resolution and the quality level of the video should be high when the motion activity is low. This is the essence of the proposed motion-based spatial-resolution adaptive WebRTC streaming system that is shown in Figure 2. We modified the WebRTC open source code[9] to include motion activity detection and motion-based layer adaptation in both scalable VP9 CBR and VBR encoding modes. A brief description of each block is provided in the following. Detailed descriptions of the novel motion activity detection and layer adaptation manager blocks are provided in Sections III.B and III.C, respectively. The remaining modules are used as is in the WebRTC open-source software.

Scalable Video Encoder: We used the VP9 video codec provided in the WebRTC platform for temporal and spatial scalable coding. We configured it for 3 spatial and 3 temporal layers. VP9 codec can perform both CBR and VBR encoding. CBR encoding mode allows fine-scale rate control. VBR coding gives a higher bitrate to the more complex segments of video while less bitrate is allocated to less complex segments. VBR is used to eliminate variable QP because we want to adapt bitrate by adapting spatial or temporal layers, not quality.

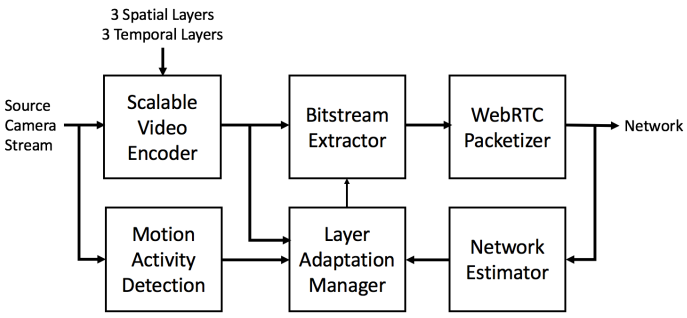

Fig. 2. The block diagram of the proposed motion-based spatial-resolution adaptive WebRTC streaming system.

Bitstream Extractor: This module extracts different spatial and temporal layers from the scalable video bitstream.

Motion Activity Detector: This is a module proposed and implemented by us to estimate a numerical measure of motion activity in the source camera stream.

Layer Adaptation Manager: This is a module proposed and implemented by us that uses motion activity measure and network bitrate estimate as inputs, and decides which spatial and temporal layers will be retained and which will be discarded for each video frame.

Network Estimator: This module implements the Google congestion control algorithm in the WebRTC open source software and also provides an estimate of available network bitrate. RTCP feedback mechanism is used to detect packet losses and the delay between packets is used to predict the available bitrate.

WebRTC Packetizer: This module uses layers that comes from the bitstream extractor and layer selection information that comes from layer adaptation manager as inputs. It packetizes the selected layers and send them to the network. Selected spatial layers are packetized as a superframe. End of superframe is set by a marker bit.

B. Motion-Activity Detection

A measure of motion activity is obtained based on the number of pixels where the frame difference between the current frame (CF) and previous frame (PF) is higher than a threshold value (DT). This number is entered into a list (VL), which keeps all values for a group of frames (GOF). We compute a weighted average of the values in this list as the motion activity measure for the GOF. The weights (WVL) are selected to emphasize the most recent frames more. Finally, the motion activity measure is compared to a selection threshold (ST) to classify the GOF as a high or low motion GOF. The complete algorithm is provided in Algorithm 1.

C. Layer Adaptation Manager

When the available network bitrate degrades, the decision between reducing spatial resolution or frame rate is made dynamically. If the motion activity is high, the number of spatial layers will decrease; otherwise, the number of temporal

Algorithm 1: Motion Activity Detection Algorithm

Input: CF, PF, VL, WVL, DT and ST
Output: $Decision$
$count$ = 0
foreach *pixel* $p_i \in CF,\ PF$ **do**
    if $|CF[p_i]$ - $PF[p_i]| > DT$ **then**
        $count$ = $count$ + 1
VL.push_front($count$)
VL.pop_back()
avg_motion = 0
foreach *index* $i \in VL,\ WVL$ **do**
    avg_motion = avg_motion + (VL[i] * WVL[i])
$Decision$ = (avg_motion $> ST$)

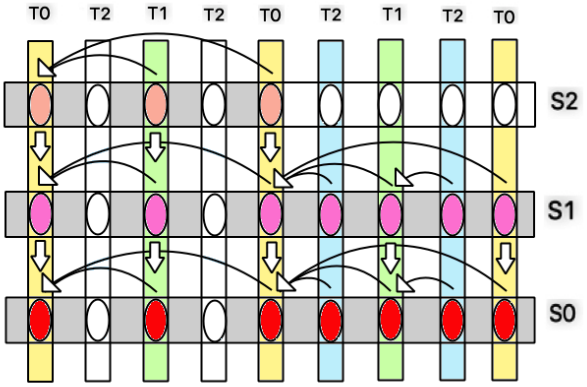

Fig. 3. Example of motion-based temporal and spatial resolution reduction for two GOFs. The yellow frames indicate the first frame of a GOF. The temporal resolution is reduced for the first GOF, while spatial resolution is reduced for the second. The white circles show the layers that are discarded.

layers will decrease. An example of motion-based temporal and spatial resolution reduction is depicted in Figure 3.

If the value of fraction loss (Loss) is higher than 0.10, indicating potential congestion, then the number of spatial or temporal layers are reduced according to the majority vote of motion states (Motion) of the last 5 group of frames. The number of layers that are decreased depends on the difference between the bitrate of the encoded stream (EBW) and the available bitrate (BWE) as shown in Algorithm 2 (between lines 23-32).

For upscaling, if the difference between available bitrate (BWE) and encoder bitrate (EBW) is larger than a threshold value (UT) and loss fraction (Loss) is lower than 0.02, the current spatial layer (SL) or temporal layer (TL) number is incremented depending on motion state (Motion) which is shown in the Algorithm 2 between lines 10-21. If one of the layers is not at the top level, we check motion state periodically. If the decremented layer is not appropriate with motion state, then we change the layer selection dynamically, corresponds to the Algorithm 2 between lines 3-9.

In our proposed mixed spatio-temporal scalable motion-based layer selective CBR mode, we use the Google Con-

Algorithm 2: Layer Selection Algorithm

1 **Input:** $Motion$, SL, CBR, TL, $Loss$, BWE, UT and EBW
2 **Output:** SL, TL **if** $Loss$ *<0.10* **then**
3 **if** *(*$SL \neq 3$*) or (*$TL \neq 3$*)* **then**
4 **if** $Motion$ *and (*$SL \neq 1$*) and (*$TL \neq 3$*)* **then**
5 SL = SL - 1
6 TL = TL + 1
7 **else if** *(*$TL \neq 1$*) and (*$SL \neq 3$*)* **then**
8 TL = TL - 1
9 SL = SL + 1
10 **if** $Loss$ *<0.02* **then**
11 **if** BWE - EBW $> UT$ **then**
12 **if** $Motion$ **then**
13 **if** $SL \neq 3$ **then**
14 SL = SL + 1
15 **else if** $TL \neq 3$ **then**
16 TL = TL + 1
17 **else**
18 **if** $TL \neq 3$ **then**
19 TL = TL + 1
20 **else if** $SL \neq 3$ **then**
21 SL = SL + 1
22 **else**
23 **if** $Motion$ **then**
24 **if** $SL \neq 1$ **then**
25 SL = SL - 1
26 **else if** $TL \neq 1$ **then**
27 TL = TL - 1
28 **else**
29 **if** $TL \neq 1$ **then**
30 TL = TL - 1
31 **else if** $SL \neq 1$ **then**
32 SL = SL - 1
33 **if** CBR **then**
34 goto Google Congestion Control Algorithm

Algorithm 3: Google Congestion Control Algorithm

1 **Input:** $TargetBitrate$, $Loss$
2 **if** $Loss$ *<0.02* **then**
3 $TargetBitrate$ = ($TargetBitrate$ + 1000) * 1.05
4 **else if** *0.02 <*$Loss$ *<0.1* **then**
5 Do nothing
6 **else**
7 $TargetBitrate$ = $TargetBitrate$ * (1 - 0.5*Loss)

gestion Control algorithm in addition to our layer selection algorithm to better match our send bitrate to the available network bitrate. Because layer selection only gives us a staircase shaped bitrate with large steps, we adapt to small changes in the available bitrate by changing QP in a small range.

Spatial and temporal layers can be downscaled at anytime when necessary, but upscaling can only occur at certain frames according to the dependency structure of the frames. This information is available in the payload description header. Temporal upscaling is allowed after a frame with enabled switching point. For spatial upscaling, the layer frame that is not an inter-predicted frame needs to be send before we start to send higher spatial layers. If the network is available, the model forces encoder to send a key frame to increase the spatial resolution immediately.

IV. Experimental Results

A. Experimental Test-bed

Experiments were conducted over a local area network with two computers connected to each other by a switch. Both computers run WebRTC client software and one of them runs a signaling server. Clients connect to the signaling server and a P2P video session is established when one of them calls the other. In order to emulate cross network traffic, we limit available uplink capacity of one computer using the software [14]. WebRTC clients become aware of the reduction in the network bitrate by means of RTCP feedback packets through the Google congestion control algorithm and decide for an appropriate motion-based source rate adaptation model.

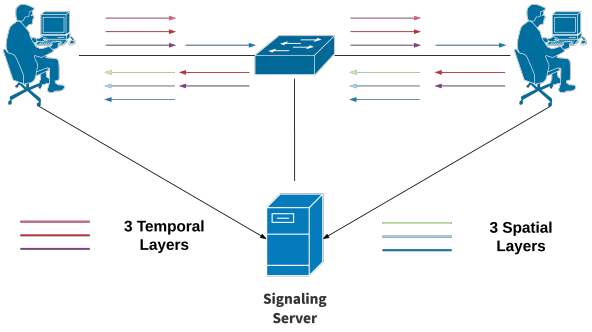

Fig. 4. Test environment.

B. Results

We conducted three experiments under identical conditions for 640x480 resolution, where we employed the default WebRTC single-layer CBR encoding, mixed spatio-temporal scalable CBR encoding with motion-based layer selection, and mixed spatio-temporal scalable VBR encoding with motion-based layer selection. The target bitrate for the default non-scalable CBR VP9 coder is set to 1 Mbps. When running the scalable VP9 encoder with three temporal and three spatial layers, the maximum video coding rate (with all layers) was set to 1.35 Mbps in the CBR mode. All experiments lasted 60 sec, where there was moderate-high subject motion up to 50 sec and very limited motion between 50 and 60 sec. We limited the available network bitrate to 600 kbps at 27 sec using the netlimiter software to compare the response of WebRTC clients with three different rate control models to this limited available bitrate. In order to select reasonable QP parameters for the adaptive VBR model, we conducted some off-line coding tests for the cases of low spatial resolution and high QP encoding with high motion video under limited available bandwidth.

The results of the experiments are shown in Figure 5, where the sent video bitrate, sent frame rate, motion activity measure, the QP value, and the number of spatial layers sent are depicted for all three scenarios. We see that all three rate control models perform similarly for the first 27 sec (where the network rate is high enough) except the bitrate with VBR mode varies with the motion more than others. We also observe that the bitrate for scalable coding options is about 30% higher than the default non-scalable coding option due overhead of scalable coding when sending all layers. When the available network rate drops to 600 kbps, the default CBR model keeps the frame rate high but increases QP value to adapt to the network rate. In the proposed motion-based spatial-resolution adaptive CBR model, we see that the number of spatial layers sent drops from 3 to 2 (320x240) at 27th second in response to dropping network bitrate while the motion activity is still high, but at 50th second it increases back to 3 when the motion activity becomes very low. The motion-based spatial-resolution adaptive VBR model behaves very similar to the proposed spatial-resolution adaptive CBR model except that it achieves a slightly lower frame rate due to coarser rate control.

In terms of visual quality, we observe that video interpolated from low spatial resolution layers introduce some blurring, which is subjectively better than video encoded at the same bitrate at full spatial resolution but with a high QP, which shows blocking artifacts. We show a representative frame that is interpolated from two spatial layers in Fig. 6, while Fig. 7 shows the same frame encoded using the default CBR model at the same frame rate with full spatial resolution but with a high QP, which shows inferior quality.

V. Conclusions

This paper shows that scalable video coding is beneficial not only for multi-party but also point-to-point WebRTC sessions. High motion activity causes higher encoding (source) bitrates. The default WebRTC CBR rate control (non-scalable encoder) increases the quantization parameter and/or reduce frame rate to maintain the desired source bitrate. For better subjective video quality, it is important to maintain a high frame rate in the presence of high motion in order to avoid motion jitter. This means we have the choice of a tradeoff between coarser quantization, which results in blocking artifacts versus spatial resolution reduction, which results in some blurring. Our results indicate that the proposed motion-based spatial-resolution adaptive rate control model achieves sufficiently high frame rates and reasonable quantization parameter values which yields more pleasing video quality compared to the default WebRTC rate control scheme, which uses only quantization parameter and frame rate for rate control.

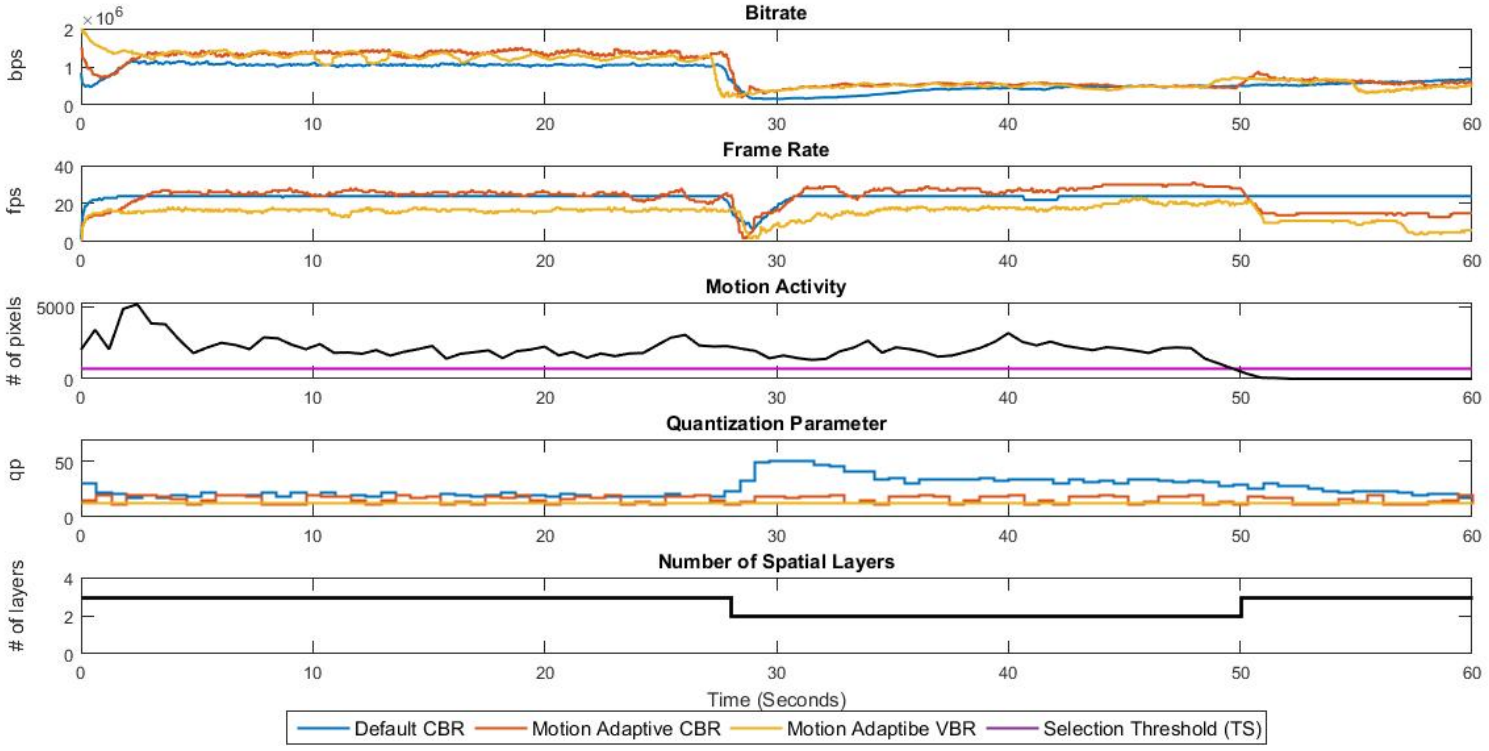

Fig. 5. Comparison of default WebRTC CBR coding, motion-based spatial-resolution adaptive CBR, and motion-based spatial-resolution adaptive VBR coding.

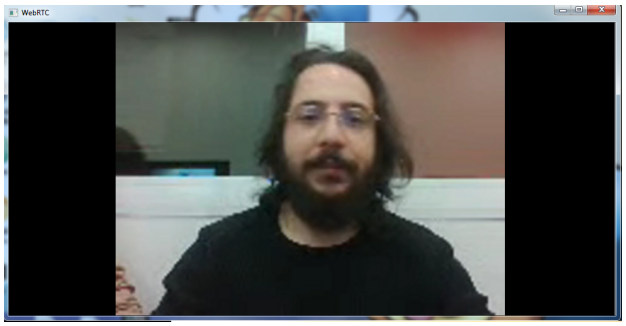

Fig. 6. Visual video quality for rate control by spatial resolution reduction.

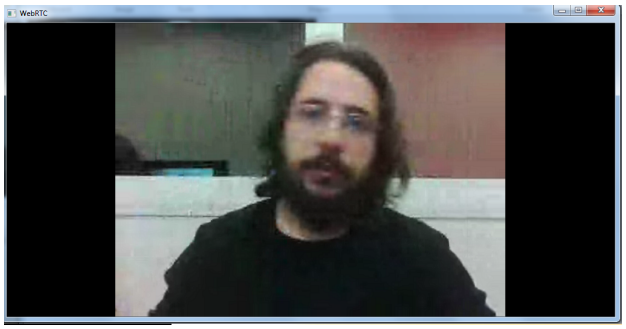

Fig. 7. Visual video quality for rate control by increasing QP.

ACKNOWLEDGMENT

This work has been funded by TUBITAK project 115E299. A. Murat Tekalp also acknowledges support from Turkish Academy of Sciences (TUBA).

REFERENCES

[1] Y. Xu, C. Yu, J. Li, and Y. Liu, Video telephony for end-consumers: Measurement study of Google+, iChat, and Skype," IEEE/ACM Trans. on Networking, vol. 22, no. 3, pp. 826839, Jun. 2014.
[2] V. Singh, A.A. Lozano, and J. Ott, Performance analysis of receive-side real-time congestion control for WebRTC," Packet Video Workshop, San Jose, CA, pp. 1-8, 2013.
[3] G. Carlucci, L. De Cicco, S. Holmer, and S. Mascolo, Analysis and design of the Google congestion control for web real-time communication (WebRTC)," Proc. 7th Int. Conf. on Multimedia Systems, 2016
[4] IETF RFC 6386, VP8 Data Format and Decoding Guide, Nov. 2011.
[5] ITU-T Rec. H.264, Advanced video coding for generic audiovisual services (V9)," February 2014. http://www.itu.int/rec/T-REC-H.264
[6] A. Grange, P. de Rivaz, and J. Hunt, VP9 Bitstream and Decoding Process Specification, 31 March 2016. https://storage.googleapis.com/downloads.webmproject.org/docs/vp9/vp9-bitstream-specification-v0.6-20160331-draft.pdf
[7] H. Schwarz, D. Marpe, and T. Wiegand, Overview of the scalable video coding extension of the H.264/AVC standard, IEEE Trans. on Circ. and Syst. for Video Tech. vol. 17, no. 9, pp. 11031120, 2007.
[8] A. Eleftheriadis, M.R. Civanlar, and O. Shapiro, Multipoint videoconferencing with scalable video coding," Jou. of Zhejiang University SCIENCE A, vol. 7, no. 5, pp. 696-705, 2006.
[9] WebRTC source code https://chromium.googlesource.com/external/webrtc/
[10] P. Baccichet, T. Schierl, T. Wiegand, B. Girod, Low-delay Peer-to-Peer Streaming using Scalable Video Coding, Packet Video Workshop, 2007.
[11] J. Liu, Y. Cho, Z. Guo, and C.J. Kuo, Bit allocation for spatial scalability coding of H.264/SVC with dependent rate-distortion analysis, IEEE Trans. Circ. and Syst. Video Tech., vol. 20, no. 7, pp. 967-981, Jul. 2010.
[12] X. Jing, J. Y. Tham, Y.Wang, K. H. Goh, andW. S. Lee, Efficient rate-quantization model for frame level rate control in spatially scalable video coding, IEEE Int. Conf. on Networks (ICON), pp. 339343, Dec. 2012.
[13] B. Hosking, D. Agrafioti, D. Bull, and N. Easton, An adaptive resolution rate control method for intra coding in HEVC, Proc. IEEE ICASSP 2016.
[14] NetLimiter, available online "https://www.netlimiter.com/"
[15] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," RFC 3550, July 2003