# Motion-Based Rate Control Using Scalable VP9 Video Coding for WebRTC Videoconferencing

by

**Gonca Bakar**

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Electrical and Electronics Engineering

KOÇ
UNIVERSITY

January 15, 2018

**Motion-Based Rate Control Using Scalable VP9 Video Coding for WebRTC Videoconferencing**

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

**Gonca Bakar**

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Prof. A. Murat Tekalp

_____

Assoc. Prof. Öznur Özkasap

_____

Asst. Prof. Ali Cengiz Beğen

Date: _____

*Dedicated to my family and friends...*

# ABSTRACT

WebRTC has become a popular platform for real-time communications (RTC) 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 constant bitrate (CBR) video encoding given the available network rate with rate control achieved by varying the quantization parameter (QP) and video frame rate. Recently, some open-source WebRTC platforms provided support for VP9 encoding with spatial scalable encoding option. This thesis seeks to incorporate motion-based spatial resolution adaptation for adaptive rate control. For point-to-point solutions the use of proposed motion-based mixed spatio-temporal scalable VP9 encoding and single-layer (non-scalable) VP9 encoding and for multi-party RTC solutions scalable VP9 video coding based Selective Forwarding Unit (SFU) with and without motion-based layer selection are compared in the presence of network congestion. Our results show that spatial resolution reduction with sufficiently high frame rates and reasonable quantization parameter values yields more pleasing video quality during intervals of high motion activity, compared to the standard rate control scheme employed in open-source WebRTC implementations that use only quantization parameter and frame rate for rate control.

# ÖZETÇE

WebRTC, en iyi çabalı internet üzerinden gerçek zamanlı iletişim uygulamaları için yaygın bir platform haline gelmiştir. Ağ trafik sıkışıklığı kontrol algoritması, ağın anlık durumu hakkında tahminlerde bulunmak için kullanılmaktadır. Varsayılan yapı, sabit bit hızlı ve tek katmanlı video kodlayıcı kullanır. Video kodlayıcı tahmin edilen uygun ağ bit hızını hedefler ve bu hedefi tutturmak için çerçeve frekansı ve nicemleme parametresini değiştirir. Son dönemlerde bazı açık kaynaklı WebRTC platformları SVC-VP9 kodlamayı desteklemeye başlamıştır. Bu tezin temel katkısı ağ sıkışıklığı durumunda video yayın hızının harekete dayalı adapte edilmesidir. Önerdiğimiz modelin uçtan uça uygulaması tek katmanlı kodlama ile, çok kullanıcılı sunucu tabanlı uygulaması ise harekete dayalı adaptasyon yapmayan modelle karşılaştırılmıştır. Sonuçlar yüksek hareket içeren aralıklarda yeterli çerçeve frekansı ve kabul edilebilir nicemleme parametresi ile çözünürlüğü düşürmenin standart hız kontrol modeline göre daha yüksek görsel video kalitesi sağladığını göstermiştir.

# ACKNOWLEDGMENTS

I would first like to express my special gratitude and thanks to my advisor, Prof. A. Murat Tekalp for his guidance and inspiration. I am also grateful to my thesis committee, Assoc. Prof. Öznur Özkasap and Asst. Prof. Ali Beğen for their valuable comment and feedbacks.

I am grateful to Arda for his feedback, cooperation and of course friendship. I would like to thank my lab mates Ogün, Selin, Serkan, Kemal, Zana, Tolga, Tuğtekin, Berker for their assistance and sharing during my graduate studies.

I also thank to my second family, Seryal, Hilal, Elif, Şahin, Küçükaşk, Deniz, my soul sister Özge and my oldest friends Yelinay, Gizem. My lovely thanks goes to Haydar for his support and encouragement during my studies.

Finally, last but not the least, I would like to thank my family for their love, patience and support.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

Chapter 1

# INTRODUCTION

## *1.1 Motivation*

Video telephony or videoconferencing over the open Internet or Web has become an essential means of real-time communications (RTC). In real time communication, negligible latency is essential. In the case of videoconferencing, to fulfill this requirement can be challenging and problematic because of the high bandwidth usage of video and limited Internet capacity. There are a number of popular proprietary solutions available to achieve reasonably acceptable delay. A comparative study of some of these solutions can be found in [1]. More recently, WebRTC has emerged as an open source 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, 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. This deficit results in queuing delays, packet loss, and delay jitter. While seconds of buffering delay is tolerable in unidirectional video streaming, in interactive RTC the maximum tolerable delay is limited to a couple of hundreds of milliseconds. 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 and scalable coding considering their various use cases and needs. Multi-party video-conferencing, where clients have heterogeneous devices and/or network connections, based on scalable video coding (SVC) [7] has been proposed and implemented for several years [8]. It is well-known that spatial scalable coding, when sending all layers, introduces a source bitrate overhead, which is approximately 25-30%, as compared to non-scalable encoding. In temporal scalability, the overhead is close to zero. However, when a client application transmits video over a congested network, packet losses and delay jitter should be taken into account. 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. 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.

In multi-part videoconferencing, a multi-point control unit (MCU) transcodes each incoming stream to multiple bitrates depending on the outgoing link capacities. But for this, an MCU needs decrypt the incoming encrypted bitstream, transcode it, an then re-encrypt to form the output bitstream. This comes at the cost of high computational power and delay. To overcome this problem, the Selective Forward Unit (SFU) architecture has been proposed for Scalable Video Coding (SVC) encoded bitstreams and implemented[9]. In an SVC based SFU, each client sends all spatial and temporal quality layers to the server. The decision of which spatial and temporal layers will be forwarded to each client is made by the SFU, depending on the resolution

of the target client screen, and available network bitrate. The server forwards different temporal and spatial video layers without decoding or re-encoding them. Hence, the advantages of scalable coding in multi-party RTC is clear.

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 re-sampling can provide a higher and more consistent level of video quality at low bitrates in the case of intra-only HEVC coding. However, none of the previous implementations consider the content of the video during video encoding rate adaptation at the client or layer selection at the SFU, which is the subject of this thesis. Motivated by these observations, this thesis 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).

### *1.2 Contributions and Organization*

To the best of our knowledge, motion-based layer selection for rate control in scalable video coding for both point-to-point (two-party) and multi-party RTC has not been studied before. We have observed that rate control implementations that do not consider the content of the video cannot achieve the best subjective video quality; and hence, spatial resolution, frame rate, and quantization parameter should be adapted to the current available network throughput by taking into account the presence of motion activity in the video.

It is well-known that the visibility of high spatial frequencies to the human eye depends on the motion activity in the video[14]. The spatial resolution and quality level of the video can be best differentiated by the human eye when the motion activity is low. This is the starting point of the proposed motion-based spatial-resolution adaptive WebRTC streaming.

The main contribution of this thesis is to propose adaptive streaming solutions using scalable VP9 video coding and motion-based layer selection for both point-to-point and multi-party RTC using an SFU. More specifically:

- We propose a point-to-point (two-party) WebRTC conferencing solution with scalable VP9 video coding and motion-adaptive layer selection at both clients. We evaluate the performance of our proposed solution versus non-scalable coding at the same bitrate in terms of received video quality at the clients in the presence of network congestion. This work has been presented in the IEEE Globecom 2017 conference [15].

- We propose a multi-party WebRTC conferencing solution with SFU using scalable VP9 video coding with motion-adaptive rate control at the clients and motion-adaptive layer selection at the SFU. We evaluate the performance of our proposed solution with another SFU solution which does not use motion information for layer selection in terms of video quality for test scenarios with different video motion and network congestion.

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.

Chapter 2

# BACKGROUND

## *2.1 Review 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 [16].

WebRTC provides three main APIs for developers: acquiring media, streaming media, and data sharing. Major APIs of WebRTC include: (i) getUserMedia, which prompts a web browser for permission to use the camera and microphone, to capture media and to represent synchronized streams of media, (ii) RTCPeerConnection, which sets up audio/video calls and handles communication between peers efficiently, and (iii) RTCDataChannel, which allows browsers to share arbitrary 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 [17]. 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 behind this structure is the limited number of IPv4 addresses. There

are 232 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.

Congestion control for real-time video conferencing is important but challenging mechanism. Efficient usage of available bandwidth is critical to avoid packet losses and provide convincing user experience. A working group, IETF RTP Media Congestion Avoidance Techniques (RMCAT) aims at developing congestion control and avoidance mechanism to be used for real-time media flows over RTP[18].

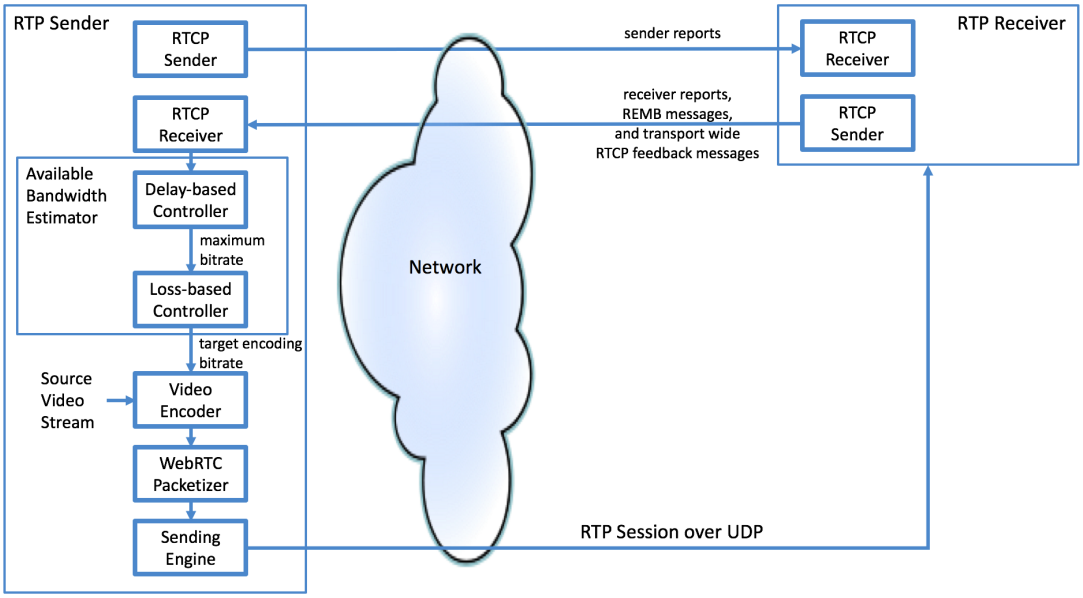


**Figure 2.1:** Congestion Control Mechanism

Figure 2.1 demonstrates work flow of congestion control modules in WebRTC. Loss-based controller and delay-based controller aim to detect congestion and to decide the target sending bitrate of packetized video stream in WebRTC:

- RTP receiver marks the time of arrival of the packet and passes it to delay-based controller either directly or through the feedback received by the RTP

sender. Using information of packets arrival time, maximum available bitrate is estimated and passed to the loss-based controller.

- Loss-based controller takes measured loss rate, measured round trip time, REMB messages, and estimated maximum bitrate and computes a target sending bitrate.

The rate is controlled using the bandwidth estimates based on loss and delay. Controllers increase the estimate of the available bandwidth until congestion is detected. After detection of over use, available bandwidth estimate is decreased by the delay-based controller. By this recursive way, available bandwidth estimate will match the real available bandwidth.

The delay based rate control subsystem has 3 states defined by IETF RMCAT working group: Increase, Decrease and Hold.[18] When there is no detected congestion, the state is Increase and the subsystem starts in this state until the detection of over-use or under-use; the state is Decrease when congestion is detected, and Hold state is for waiting before going to Increase state. Table 2.1 shows the state transitions.

**Table 2.1:** State Transitions

| **Signal\State** | **Hold** | **Increase** | **Decrease** |
|---|---|---|---|
| **Over-use** | Decrease | Decrease | Remain in State |
| **Normal** | Increase | Remain in State | Hold |
| **Under-use** | Remain in State | Hold | Hold |

How much of the delay-based available bandwidth estimate will increase depends on the current state. If the current available bandwidth estimate is too far from convergence, estimate is increased multiplicatively, otherwise it is increased additively.

If the delay-based available bandwidth estimate increases more than its three standard deviations, the current congestion level is assumed to be changed. In this

case, the average maximum bitrate is reset and estimate returns to be increased multiplicatively.

Decisions of the loss-based congestion controller are based on the round-trip time, loss rate measurement and maximum available bandwidth which is estimated by the delay-based controller.

The maximum available bandwidth estimates based on delay is only accurate when the queues are sufficiently large. Otherwise, packet losses will be only reliable indicator of congestion.

The loss-based controller runs estimation mechanism when RTCP feedback from the receiver is received.

**Algorithm 1:** 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)
```

The loss-based available bandwidth is estimated using Algorithm 1. The input, $TargetBitrate$ is produced by delay-based controller. The loss-based estimate is compared with the one estimated by delay-based controller and the minimum between them is used as the actual sending bitrate.

### *2.2 Review of VP9/SVC*

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.

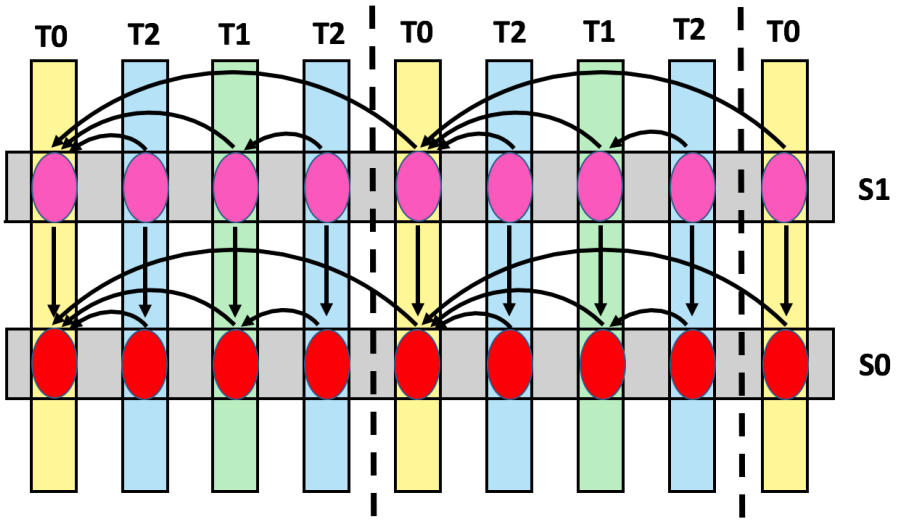


**Figure 2.2:** Mixed spatio-temporal scalable coding with two spatial (S0, S1) and three temporal (T0, T1, T2) layers and the dependency between the layers.

The high quality video bit stream which is encoded using Scalable Video Coding standards contains multiple subset bit streams. Instead of sending multiple streams with repetitive and redundant information and packetization overhead, scalable video encoding allows to send several encoded layers within the same bitstream.

A scalable video encoder produces multiple encoded bitstream layers that depend on each other with hierarchical pattern. A specific layer, together with the lower layers which it depends on, determines a particular spatial and temporal resolution. The layer that does not need any other layer to be decoded determines the lowest spatio-temporal resolution and is called the base layer. Each additional layer improves spatial or temporal quality of the video. Layer dependency structure of mixed scalable encoded video with two spatial and three temporal layers is depicted in Fig. 2.2 for two a picture groups (PG) with four pictures each. In the figure, S0 (red) and S1 (pink) denote spatial layers, while T0 (yellow), T1 (green) and T2 (blue) denote

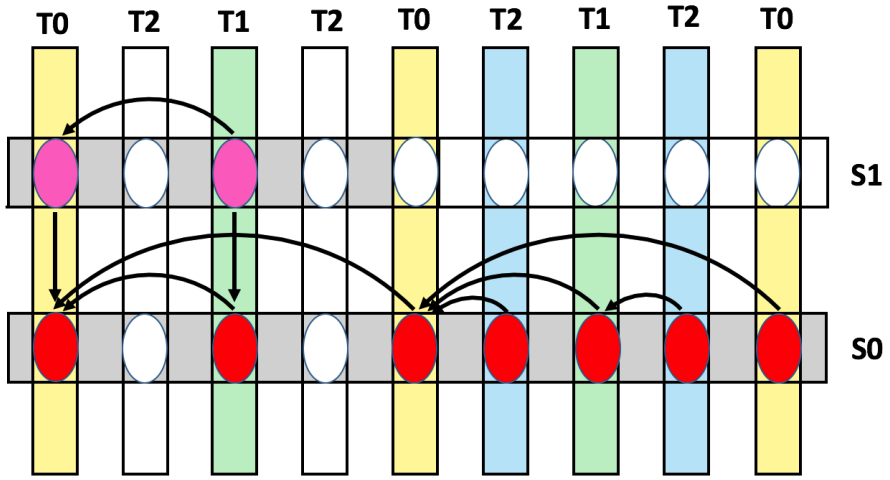


**Figure 2.3:** Example of motion-adaptive layer selection for two PGs with four pictures each. The yellow frames indicate the first frame of a PG. The temporal resolution is reduced for the first PG, while spatial resolution is reduced for the second PG. White circles show the layers that are discarded.

temporal layers. Horizontal arrows show the dependencies between temporal layers and perpendicular arrows show spatial dependencies.

A subset video bitstream with lower spatial or temporal layer than the whole bitstream is derived by discarding packets. Fig. 2.3 depicts an example of temporal and spatial resolution reduction. White circles show the layers that are discarded and frames shown in this figure can be decoded at spatial and temporal layer of two.

VP9 manages the spatial scalability structure by using super frames. A super frame consists of one or more frames from different spatial layers. All lower spatial layers that one layer frame depends on and this layer frame are included in the same super frame. In this way, reduction of the upper layers does not destroy the hierarchy between frames.

There are two types of payload formats for a scalable VP9 stream: flexible mode and non-flexible mode. In the flexible mode, it is possible to change number of layers and their hierarchical patterns dynamically. In the non-flexible mode, the structure of layer dependencies within a picture group (PG) are pre-specified as part of the scalability structure (SS) data. SS data is sent once for each PG and gives information

about the resolution of different spatial layer. In non-flexible mode, each layer frame must have an spatial layer index.

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 adapting the quantization parameter value (quality level adaptation).

## *2.3 Review of Multi-party Videoconferencing Design Architectures*

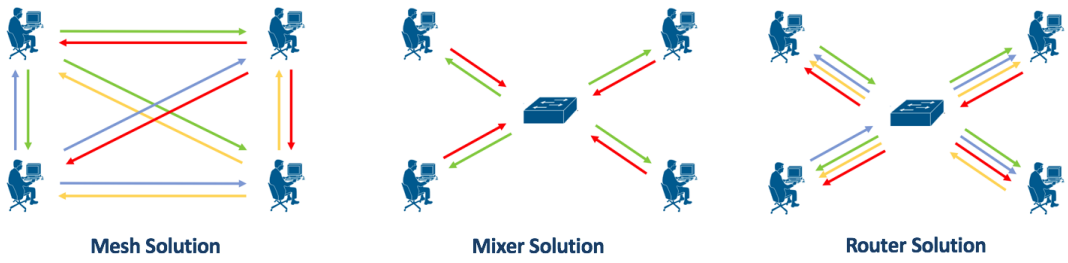


**Figure 2.4:** Multi-party videoconferencing design architectures.

Video conferencing is a communication session between two or more parties who can watch and hear each other by means of two-way transmission of video and audio regardless of their different locations in near real-time. Many use cases such as e-learning, customer support, communication within companies and patient care require video communication. Simple point-to-point conversation between two persons, or a multi-point conference between many at different locations can satisfy the need of application. For multi-party video conferencing, different architectures can be categorized as: peer-to-peer (P2P) and centralized[19]. In full-mesh P2P conferencing, each client needs to connect all other clients directly. On the other hand, centralized-servers can be used for mixing or routing solutions for multi-party video conferencing.

### 2.3.1 Mesh solution

The Mesh P2P solution is the simplest architecture for multi-party videoconferencing. This solution is based on generating multiple point-to-point streams from each participant to all other participants. Despite its simplicity, the direct connection between participants requires to high bandwidth usage, making it impractical. As a result, architecture with some medium is a more effective option.

### 2.3.2 Mixer solution

The Mixer approach has been used for years with great success in videoconferencing. This architecture is an example of centralized solution. A central unit maintains one-to-one streams between each participant. The Multi-point Control Unit (MCU) creates one outgoing media stream for each participant by mixing incoming media streams. In contrast with the other solutions, endpoints receive only single incoming media stream with mixer architecture. Hence, mixer solution provides the least complexity for endpoints.

Full bitrate adaptation in MCU is achieved by decoding each incoming stream, changing their frame rate and/or resolution, encoding and sending them. It is efficient in terms of bandwidth utilization because the server can create media streams with different resolution, frame rate or quality for each endpoint. But on the other hand, this feature of MCU may turn into the main drawback of this solution. Because the operation of decoding and re-encoding requires massive computational power and introduces extra processing delay. Quality may decrease as a result of lossy re-encoding operation. Also, transcoding and composition may result in less flexibility for the user interface of the application and less scalability.

### 2.3.3 Router solution

In addition to mixer solution, routing is another centralized solution for video conferencing systems. Unlike MCU, Selective Forwarding Unit (SFU) does not decode

and re-encode the media but sends the copies of the required parts of the incoming streams to other participants. Each endpoint receives one stream for each other participant. If the video is not scalable encoded, these incoming streams will be in full quality. When the endpoint is not capable of displaying incoming media streams simultaneously in full resolution, it needs to reduce the spatial quality on its side which is inefficient for bandwidth utilization.

The Router architecture became more common with the developing of H.264 SVC standard. Scalable VP9 encoding constitutes a good alternative for H.264 SVC because it comes without any legacy baggage. This approach is based on a central unit receiving an scalable coded media stream from each participant. The router cuts off excessive layers and sends out video stream of different quality to every participants. SVC stream varies at most 30% in the bandwidth from non-SVC stream, but this approach compensates for this overhead by preventing unnecessary layers from being sent.

Selective Forwarding Unit (SFU) generates different sets of streams for each endpoint with respect to their available bandwidth, computational resources, screen resolution etc. SFU only does packet extraction and selective forwarding, but not computationally expensive decoding and re-encoding operations. Usage of SFU with scalable coding provides a computationally cheap and scalable multi-party approach, with less delays and flexibility.

Chapter 3

# POINT-TO-POINT RTC

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

## *3.1 System Architecture*

The bitrate of the mixed scalable coded source video must adapt dynamically 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.

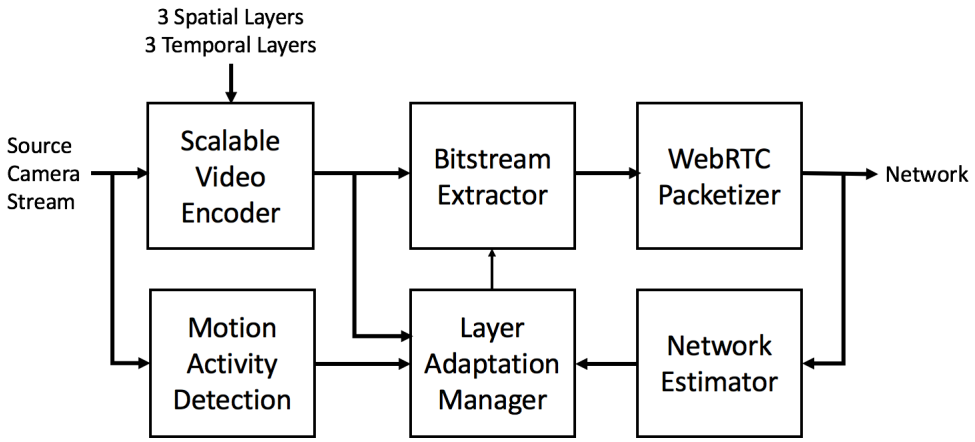


**Figure 3.1:** The block diagram of the proposed motion-based spatial-resolution adaptive WebRTC streaming system.

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. This defect affects user experience negatively. High spatial frequencies are not visible to human eye when the motion activity is 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. Fig. 3.1 shows the block diagram of this framework. We modified the WebRTC open source code[16] 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:

*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. A marker bit sets the end of the superframe.

### *3.2 Motion-Activity Detection*

For real-time communication, real time operation to achieve low delay is more important than a highly accurate detection algorithm. Two or three level coarse motion activity detection is sufficient for layer selection. That is why we use simple algorithm for motion detection rather than a more sophisticated one. A measure of motion activity is obtained for each picture group based on the number of pixels, where the frame difference between the current picture $CF$ and previous picture $PF$ is higher than a threshold value, $T_D$. For each picture in the picture group, this number is entered into a list, $VL$. A binary (low, high) or ternary (low, medium, high) motion activity measure is computed as a function $f(\cdot)$ of the weighted average $avg_mot$ of the values in this list. The weights, $WVL$ are selected to emphasize the more recent frames. The complete algorithm is provided in Algorithm 2.

### *3.3 Layer Adaptation Management*

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. If the motion activity measurement is below the threshold, the number of temporal layers will decrease. Figure 2.3 depicts an example of motion-based temporal and spatial resolution reduction.

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 five group of frames. The number of layers that are decreased depends on the difference between the bitrate of the encoded stream ($BR$) and the available bitrate ($BW_E$) as shown in Algorithm 3 (between lines 26-36).

**Algorithm 2:** Motion Activity Detection

```
1  Input: Picture Group, WVL, T_D
2  Output: Motion
3  foreach CF ∈ Picture Group do
4      if CF = first frame of Picture Group then
5          PF = CF
6          break
7      count = 0
8      foreach pixel p_i ∈ CF, PF do
9          if |CF[p_i] - PF[p_i]| > T_D then
10             count = count + 1
11     VL.push_front(count)
12     VL.pop_back()
13     PF = CF
14 avg_mot = 0
15 foreach index i ∈ VL, WVL do
16     avg_mot = avg_mot + ( VL[i] * WVL[i] )
17 Motion = f(avg_mot)
```

For upscaling, if the difference between available bitrate, $BW_E$ and encoder bitrate, $BR$ is larger than a threshold value, $T_U$ 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 3 between lines 14-25. 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 3 between lines 6-13.

In our proposed mixed spatio-temporal scalable motion-based layer selective CBR mode, QP varies in a reasonable range in addition to the layer selection algorithm to better match the send bitrate to the available network bitrate. Because layer selection only gives us a stair-case 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

**Algorithm 3:** Layer Selection Algorithm

```
1  Input: Video, Motion, BR, T_U
2  Output: SL, TL
3  foreach Picture Group do
4      Loss ← Google Congestion Control Algorithm
5      BW_E ← Google Congestion Control Algorithm
6      if Loss <0.10 then
7          if (SL ≠ 3) or (TL ≠ 3) then
8              if Motion and (SL ≠ 1) and (TL ≠ 3) then
9                  SL = SL - 1
10                 TL = TL + 1
11             else if (TL ≠ 1) and (SL ≠ 3) then
12                 TL = TL - 1
13                 SL = SL + 1
14         if Loss <0.02 then
15             if BW_E - BR >T_U then
16                 if Motion then
17                     if SL ≠ 3 then
18                         SL = SL + 1
19                     else if TL ≠ 3 then
20                         TL = TL + 1
21                 else
22                     if TL ≠ 3 then
23                         TL = TL + 1
24                     else if SL ≠ 3 then
25                         SL = SL + 1
26     else
27         if Motion then
28             if SL ≠ 1 then
29                 SL = SL - 1
30             else if TL ≠ 1 then
31                 TL = TL - 1
32         else
33             if TL ≠ 1 then
34                 TL = TL - 1
35             else if SL ≠ 1 then
36                 SL = SL - 1
```

the frames. The payload description header gives the information of the suitability for upscaling. Switching point header enables temporal upscaling of the following frame.

For spatial upscaling, the layer frame that is not an inter-predicted frame needs to be send before sending higher spatial layers. If the available bandwidth can afford, the model forces encoder to send a key frame to increase the spatial resolution immediately.

## *3.4 Experimental Results*

### *3.4.1 Experimental Test-bed*

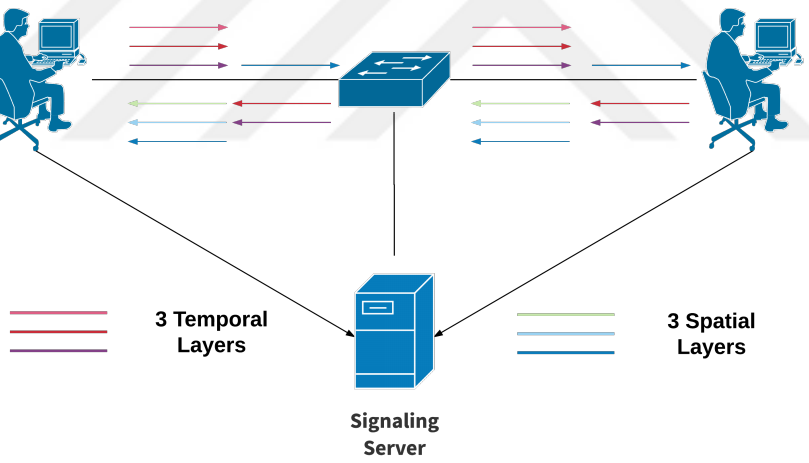


**Figure 3.2:** Test environment of point-to-point RTC.

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 [20]. 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.

### 3.4.2 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. Scalable coded videos have three spatial layers with 640x480, 320x240, and 160x120 resolution and three temporal layers with 30, 15, and 7 fps. 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 seconds, where there was moderate-high subject motion up to 50 seconds and very limited motion between 50 and 60 seconds. We limited the available network bitrate to 600 kbps at 27 seconds using the network limiter 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 off-line coding tests for the cases of low spatial resolution and high QP encoding with high motion video under limited available bandwidth.

Figure 3.3 shows the results of the experiments, 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 observed that all three rate control models perform similarly for the first 27 seconds (where the congestion is not occurred yet) except the bitrate with VBR mode varies with the motion more than others. We also observed 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.

We observed that video interpolated from low spatial resolution layers introduce some blurring. On the other hand, video encoded at full spatial resolution but with a high QP shows blocking artifacts. In terms of visual quality, blurring is subjectively better than blocking.

Fig. 3.4 and Fig. 3.5 show a representative frames. Frames on the left are interpolated from two spatial layers and frames on the right are encoded using the default CBR model at the same frame rate with full spatial resolution but with a high QP, which shows inferior quality. Figures indicate that the proposed motion-based spatial-resolution adaptive rate control model which achieves more pleasing video quality compared to the default WebRTC rate control scheme.

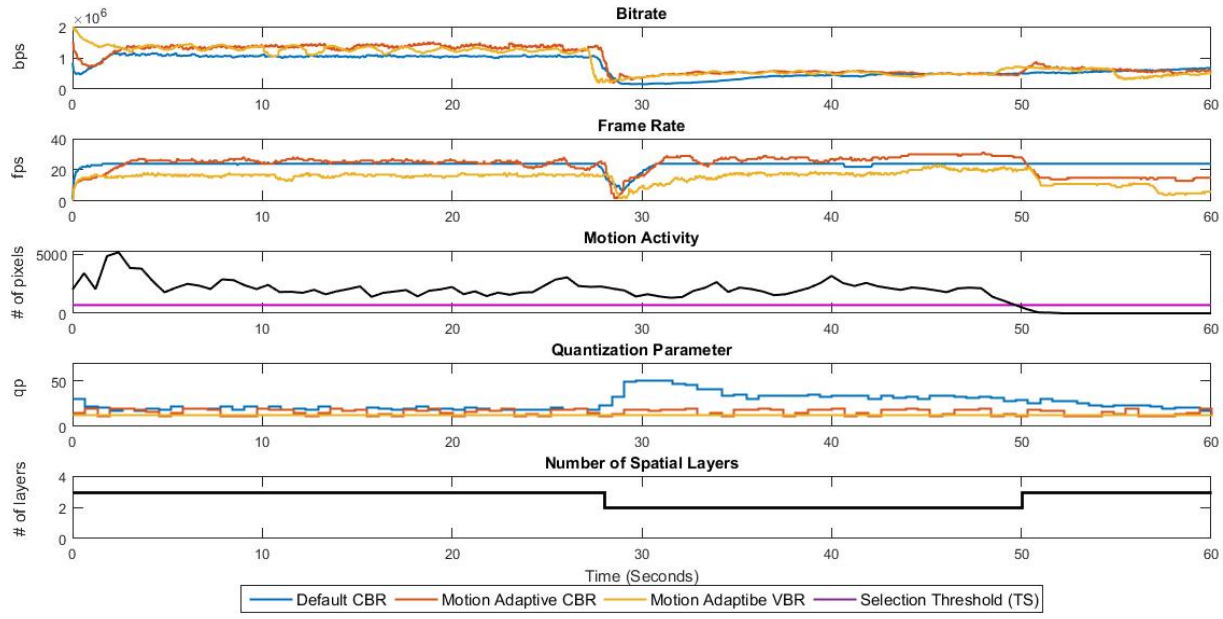


**Figure 3.3:** Comparison of default WebRTC CBR coding, motion-based spatial-resolution adaptive CBR, and motion-based spatial-resolution adaptive VBR coding.

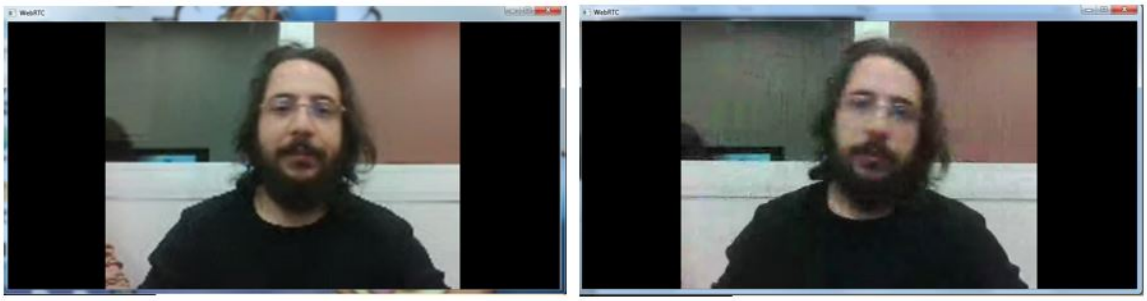

**Figure 3.4:** Visual video quality for rate control by increasing QP and by spatial resolution reduction.

**Figure 3.5:** Visual video quality for rate control by increasing QP and by spatial resolution reduction.

Chapter 4

# MULTI-PARTY RTC

This chapter first describes the proposed system architecture for motion-based mixed spatio-temporal resolution rate control in multi-party video conferencing. We then provide the details of rate control at the clients and the proposed motion-adaptive layer selection method at the SFU. Finally, we present the experimental results.

## *4.1 System Architecture*

The bitrate of the mixed scalable coded video must be dynamically adapted to the time-varying network bitrate both from a client to the SFU and from the SFU to all other clients. When sending a video stream from a client to the SFU, the adaptation of the video bitrate can be achieved by changing the frame rate, spatial resolution, or quality (increasing quantization parameter) at the encoder. When forwarding incoming video streams from the SFU to clients, the total bitrate of all streams going to a particular target client must not exceed the estimated available bandwidth to that client to avoid delay jitter and packet losses. As the network bitrate from the SFU to a client varies, the adaptation of the total bitrate can be achieved by only selecting the number of temporal and/or spatial layers of each video stream in proportion to their full layer bitrates and motion contents.

The spatial resolution and frame rate must be related to the amount of motion activity, such that the spatial resolution of the video should be high when the motion activity is low, and the frame rate must be high when motion activity is high. This is the essence of the proposed motion-based spatio-temporal resolution adaptive WebRTC streaming system. The block diagram of the proposed system is depicted in Figure 4.1. A brief description of each block is provided in the following:

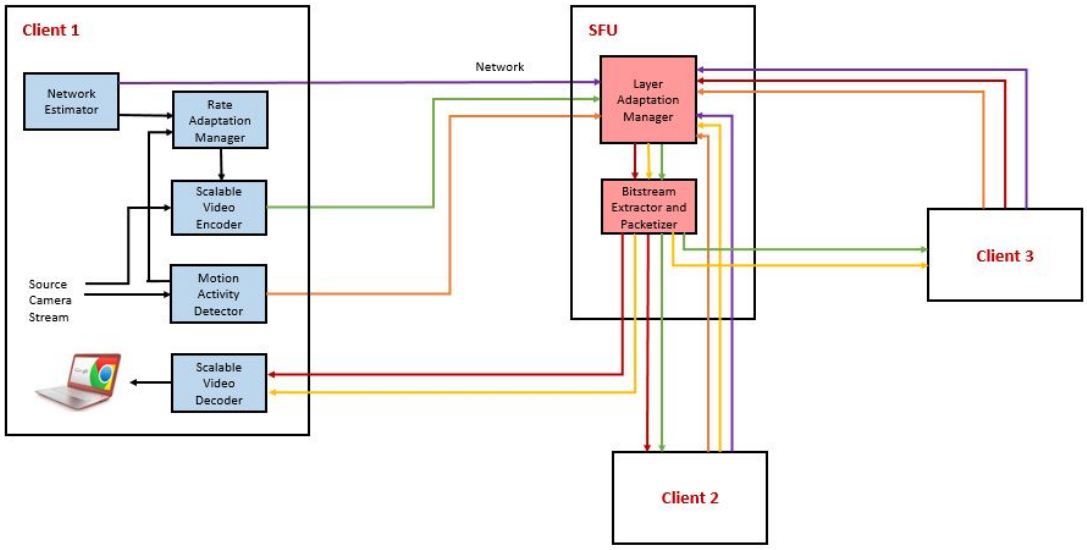


**Figure 4.1:** The block diagram of the proposed motion-based spatial-resolution adaptive WebRTC streaming system. The blocks for Client 2 and Client 3 are identical to that of Client 1.

*Scalable Video Encoder:* This module encodes the source video into a single bitstream with a specified full-layer bitrate and number of spatial and temporal layers.

*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.

*Motion Activity Detector:* This is a module, which we implemented to estimate a numerical measure of motion activity in the source camera stream at each client. The motion activity of the video is categorized as low, moderate or high for each picture group (PG) according to some selected thresholds.

*Rate Adaptation Manager:* This module takes the motion activity measure and network bitrate estimate as inputs, and decides the resolution of spatial layers and the frame rate to be used by the scalable video encoder at the client.

*Scalable Video Decoder:* The decoder generates the video stream with embedded spatial and temporal layers from the scalable encoded bitstream.

*Layer Adaptation Manager:* This is a module proposed and implemented by us that takes the full-layer bitrates and motion activity measure for all incoming client video streams and the network bitrate estimate for each target client as inputs, and decides which spatial and temporal layers will be forwarded to each target client.

*Bitstream Extractor and Packetizer:* This module takes video streams for all clients and layer selection information for each target client that comes from the layer adaptation manager as inputs. It extracts the selected layers for each target client from the scalable video stream, packetizes and sends them to the network.

## *4.2 Video Rate Adaptation at Clients*

This section provides a detailed explanation of how the spatio-temporal resolution of the source video at the clients are adapted given the time-varying network bitrate estimate from a client to the SFU and the motion activity of the source video.

The bitrate of the mixed spatio-temporal scalable coded video must be adapted dynamically according to the changing network bitrate estimate. This can be achieved by dynamically varying the spatial resolution and/or frame rate, and the quantization parameter considering the amount of motion in the video.

There are two options in adapting the rate of video at clients: i) Encode scalable video (with all layers) at a fixed target bitrate with fixed spatial and temporal layer resolutions, and drop some enhancement layers that are sent to the SFU as the network rate fluctuates, or ii) adapt the target encoding rate, and possibly the spatial and temporal layer resolutions, for the scalable video for all layers, and always send all layers to the SFU. These two options have different advantages and disadvantages.

The former does not require re-initiation of the encoder. However, in order to increase the number of spatial layers, the encoder needs to consider the dependency structure of frames, which typically results in delayed adaptation of the bitrate. Furthermore, dropping layers at the client encoder provides only coarse-level bitrate adaptation and limits the ability of the SFU for rate adaptation while forwarding.

The latter approach requires re-initialization of the encoder at the client when varying the spatial resolution of the full-layer video, which may incur some bitrate overhead. On the other hand, it allows fine granular bitrate adaptation allowing full utilization of the available network bandwidth between the client and SFU, and does not limit the bitrate adaptation flexibility at the SFU.

Considering the advantages and disadvantages of both options, we have chosen to implement the latter approach. In addition to adapting the target bitrate of the full-layer video to match the available network bandwidth, we adapt the spatial resolution and/or frame rate of the video to achieve the best visual quality for a given bitrate according to Algorithm 4. The encoder is always configured with 2 spatial and 3 temporal layers. The temporal scalability layers are set for 30, 15 and 7,5 fps, respectively. However, the organization of the temporal enhancement layer bitstream enables dropping frames instantaneously allowing intermediate values of fps to be achieved. This does not affect the temporal layer adaptation at the SFU.

The inputs to Algorithm 4 are the estimated bandwidth $BW_{target}$ between the client and the SFU, thresholds $T_{BW1}$ and $T_{BW2}$ to determine high and low available bandwidth, respectively, current value of the quantization parameter $QP$, current frame rate $FrRate$, current spatial resolution $SplRes$, maximum and minimum spatial resolution levels $Res_{max}$ and $Res_{min}$, respectively, and motion activity measure $Motion$. The Algorithm outputs the optimal spatial resolution $NewSplRes$ and frame rate level $NewFrRate$ according to values of $QP$, $BW_{target}$, and $Motion$.

In our system, we use three spatial resolution levels: high, medium and low, where high resolution refers to 480p, medium resolution is 360p and low resolution is 240p. We also quantize QP and frame rate into three ranges, low, medium and high. The low frame rate corresponds to 1-10 fps, medium frame rate refers to 11-20 fps, and high frame rate corresponds to 21-30 fps. The QP ranges are set according to the level of visual artifacts. Low QP values result in no observable visual artifacts, e.g., QP less than 20. Medium QP range, e.g., 20-30, results in minor visual artifacts, and high QP range, e.g., greater than 30, results in observable visual artifacts.

**Algorithm 4:** Video Encoding Rate Adaptation at Clients

**Input:** $BW_{target}$, $T_{BW1}$, $T_{BW2}$, $QP$, $FrRate$, $SplRes$, $Res_{max}$, $Res_{min}$, $Motion$
**Output:** $NewSplRes$, $NewFrRate$
**foreach** *Picture Group* **do**
  **if** *($BW_{target} \geq T_{BW1}$) and ($QP = low$)* **then**
    **if** *($Motion = high$)* **then**
      **if** $FrRate \neq high$ **then**
        $NewFrRate \leftarrow$ increase $FrRate$
      **else if** *($SplRes \neq Res_{max}$)* **then**
        $NewSplRes \leftarrow$ increase $SplRes$
    **else**
      **if** *($SplRes \neq Res_{max}$)* **then**
        $NewSplRes \leftarrow$ increase $SplRes$
      **else if** $FrRate \neq high$ **then**
        $NewFrRate \leftarrow$ increase $FrRate$
  **else if** *($BW_{target} \geq T_{BW2}$) and ($QP = medium$)* **then**
    **if** *($SplRes \neq Res_{max}$) or ($FrRate \neq high$)* **then**
      **if** *($Motion = high$) and ($SplRes \neq Res_{min}$) and ($FrRate \neq high$)* **then**
        $NewSplRes \leftarrow$ decrease $SplRes$
        $NewFrRate \leftarrow$ increase $FrRate$
      **else if** *($Motion = low$) and ($FrRate \neq low$) and ($SplRes \neq Res_{max}$)* **then**
        $NewSplRes \leftarrow$ increase $SplRes$
        $NewFrRate \leftarrow$ decrease $FrRate$
  **else**
    **if** *($Motion = high$)* **then**
      **if** *($SplRes \neq Res_{min}$)* **then**
        $NewSplRes \leftarrow$ decrease $SplRes$
      **else if** $FrRate \neq low$ **then**
        $NewFrRate \leftarrow$ decrease $FrRate$
    **else**
      **if** *($FrRate \neq low$)* **then**
        $NewFrRate \leftarrow$ decrease $FrRate$
      **else if** *($SplRes \neq Res_{min}$)* **then**
        $NewSplRes \leftarrow$ decrease $SplRes$

If $QP$ is low and the available target bandwidth $BW_{target}$ is above the high threshold $T_{BW1}$, the algorithm checks whether spatial resolution and frame rate are equal to the possible maximum value. If not, spatial resolution and/or frame rate are increased according to the amount of the motion in the video (lines between 3-13).

If $QP$ is medium and available target bandwidth $BW_{target}$ is between the thresholds $T_{BW1}$ and $T_{BW2}$, that means the current $QP$ is reasonable for this bandwidth range. In this case, algorithm controls whether the motion amount of the video has changed. If the current spatial resolution and frame rate combination is not proper for the motion level, the trade-off between spatial resolution and frame rate is adjusted (lines between 14-21).

Finally, if the target bandwidth $BW_{target}$ is limited and $QP$ is not low enough to achieve a reasonable quality, the spatial resolution and/or frame rate that is not equal to the possible minimum value is decreased regardless of the amount of motion in the video (lines between 22-32).

### *4.3 Layer Adaptation at the SFU*

Adaptation of video source encoding rate to the available network bandwidth at the client side is allowed to vary the instantaneous frame rate and quantization parameter dynamically at the encoder, which makes full utilization of continuously varying bandwidth between the client and SFU possible. On the other hand, rate adaptation at the SFU is only possible by means of spatial and temporal layer selection, which results in a stair-case type of bitrate as a function of time for the bitstream of a single client. However, in $N$-party videoconferencing the SFU must consider fitting the sum of bitrates of remaining $N-1$ clients to the available bandwidth of the target client. That is, we must adapt the sum of $N-1$ stair-case functions to best fit the continuously varying bandwidth of the target client. This is why optimal selection of scalable layers jointly for multiple users is a challenging problem in multi-party videoconferencing with SFU.

### *4.3.1 Modeling Rate of Layers*

It is important to know how much the rate of the video will change in response to layer selection. In theory, it is possible to exactly calculate the bitrate of the video for each combination of layers at the encoder. However, this information must either be sent to the SFU as bitrate overhead, or must be recalculated at the SFU on-the-fly. A more efficient approach in terms of processing overhead at SFU and delay is to model the ratio of the bitrate of each combination of layers to the bitrate of full video, and use an estimate of the bitrate value computed from the model instead of calculating the real bitrate at each decision instant.

The proposed model is based on the observation that the ratio of bitrates of a given combination of spatial and temporal layers to the bitrate of full video (all layers) is approximately constant over time for most head and shoulders type of videoconferencing videos. This is demonstrated in Fig. 4.2, which shows the bitrates of six bitstreams obtained from combinations of two spatial layers and three temporal layer of a single scalable VP9 encoded bitstream [21]. The full-layer bistream, shown in yellow, is encoded at 2 Mbps. It can be observed that the ratio of the bitrates of different layer combinations is nearly constant over time.

In the scalable encoder configuration, we have two spatial layers, $SL = 0, 1$, and three temporal layers, $TL = 0, 1, 2$. Let $W_{SL,TL}$ be our model coefficient denoting the ratio of bitrate $BW_{SL,TL}$ of the bitstream with spatial layer $SL$ and temporal layer $TL$ to that of the full-layer bitstream $BW_{1,2}$. In order to find appropriate values for our model coefficients $W_{SL,TL}, SL = 0, 1$ and $TL = 0, 1, 2$, we calculated the ratio of bitrates for all possible layer combinations for 5 test videos, and computed the mean and variances, which are tabulated in Table 4.1. Clearly, the coefficient representing the full-layer bistream $W_{1,2} = 1$ by definition, and is not shown in the table. It can be observed that the estimated variances of the ratios are less than 1% of the estimated mean values for all ratio coefficients. Hence, we conclude that the proposed model coefficients can provide accurate estimates of bitrates for different combinations of layers given the full-layer bitrate for the scalable encoded video.

**Table 4.1:** Estimated Means and Variances of Ratios

| | $W_{0,0}$ | $W_{0,1}$ | $W_{1,0}$ | $W_{1,1}$ | $W_{0,2}$ |
|---|---|---|---|---|---|
| **Mean** | 0.1694 | 0.2465 | 0.5133 | 0.7462 | 0.3362 |
| **Variance** | 4.6185e-04 | 9.7895e-04 | 0.0053 | 0.0078 | 0.0021 |

Then, the bitrate $BW_{SL_{out},TL_{out}}$ of the outgoing video for combination of spatial layer $SL_{out}$ and temporal layer $TL_{out}$ can be estimated at the SFU by

$$BW_{SL_{out},TL_{out}} = BW_{SL_{in},TL_{in}} \times \frac{W_{SL_{out},TL_{out}}}{W_{SL_{in},TL_{in}}} \tag{4.1}$$

given the bitrate $BW_{SL_{in},TL_{in}}$ of the incoming bitstream with spatial layer $SL_{in}$ and temporal layer $TL_{in}$ and the model coefficients $W_{SL,TL}$.

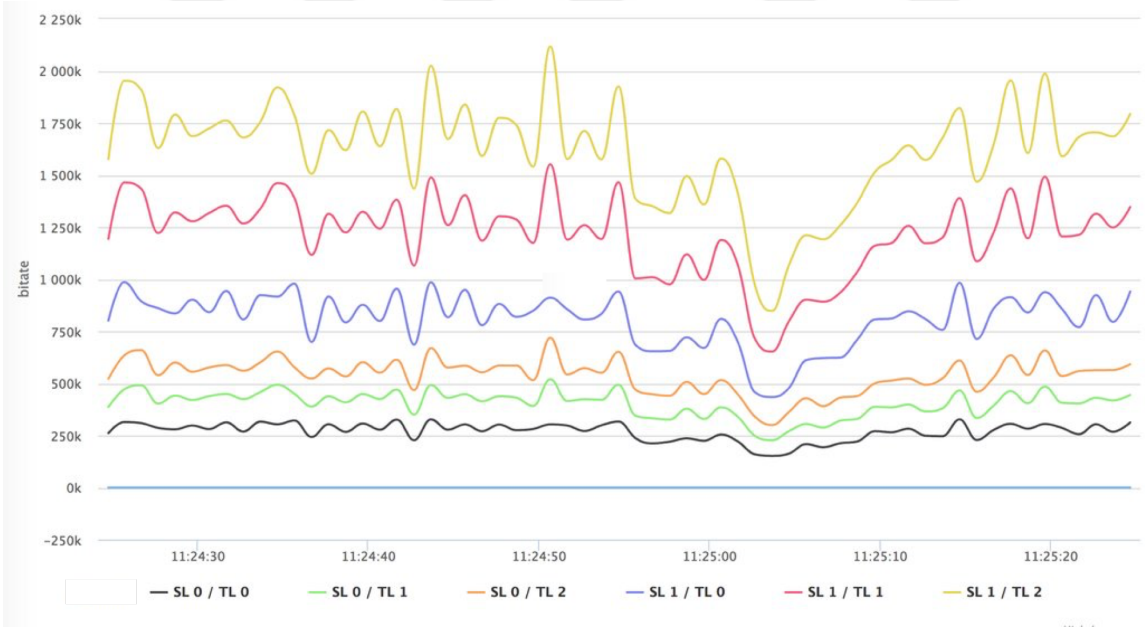


**Figure 4.2:** Bitrates of different layer combinations of scalable VP9 encoded video[21].

### *4.3.2 Formulation as an Optimization Problem*

In a multi-party videoconference with $N$ clients, where each client employs scalable video coding with 2 spatial and 3 temporal layers, the SFU is faced with deciding what is the best combination of spatial and temporal layers to forward to client $i$

given the incoming video layers from the remaining $N-1$ clients. That is, for each target client $i$, there are a total of $6^{N-1}$ combinations of layers to choose from.

Since the adaptation of the bitrate of outgoing streams to the bandwidth of the target client at the SFU can only be achieved by spatial and temporal layer selection, a perfect match to the available bandwidth may not be achievable. To this effect, we aim to minimize the difference between the estimated available bandwidth to target client $i$ and the estimated total bitrate of outgoing streams for each outgoing target client $i$, using our model in Section 4.3.1. This bitrate difference is defined by

$$\Delta(i, count) = BW_{Target}(i) - \sum_{j=0, j\neq i}^{N-1} (BW(j) \times \frac{W_{selected}(j, count)}{W_{in}(j)}) \tag{4.2}$$

where $N$ denotes the number of clients in the video-conference, $i$ denotes the target client, *count* denotes a particular combination (out of $6^{N-1}$ combinations) of spatial and temporal layers for the remaining $N-1$ clients, $BW_{Target}(i)$ denotes the estimated available bandwidth for the target client $i$, $BW(j)$ denotes the bitrate of video incoming from client $j$, and $W$ denotes approximated ratios for all spatial and temporal layers represented in a vector with size 6.

While minimizing this bitrate difference, we need to take three issues into account and introduce constraints to address them. Firstly, the total bitrate of the video streams must not exceed the estimated bandwidth to prevent the packet loss. The first constraint must avoid the situation that objective function is negative. Secondly, the optimization needs to degrade all videos fairly; hence, the second constraint must avoid the situation that the quality of one video is decreased harshly when others are kept at the highest layer. Also, we need to consider motion measures of the videos to decide which layer to decrease. The spatial and temporal layers of the video that sent from Client n to SFU are $SL_{in}(j)$ and $TL_{in}(j)$. $W_{SL_{in}(j),TL_{in}(j)}$ is known because the spatial and temporal layer information is sent periodically to SFU from each client. We are looking for optimal $SL_{out}(j)$ and $TL_{out}(j)$ that minimizes the difference for each n where $j = 0, \ldots, N-1$, $j \neq i$. For the video encoded with

2 spatial layers and 3 temporal layers, the video can be decoded to the 6 possible videos with different layer combinations. Therefore, there are 6 possible weights for full layered video. According to the amount of the motion in the video, we eliminate some possible weights that we are looking for to minimize the difference between the available bitrate and total sending bitrate. For example, for a video with high motion content, the possibilities with the lowest temporal layer are not calculated while looking for the optimal solution. The optimization problem is solved over the weights with temporal layer 1 and 2, not 0. This constraint is applied to the algorithm by eliminating weights with undesirable layers by setting them to infinity.

Then, the optimization problem for each target client $i$ can be formulated as

$$
\begin{aligned}
\underset{SL_{out},TL_{out}}{\text{minimize}} \quad & BW_{Target}(i) - \sum_{j=0,j\neq i}^{N-1} (BW(j) \times \frac{W_{SL_{out}(j),TL_{out}(j)}}{W_{SL_{in}(j),TL_{in}(j)}}) \\
\text{subject to} \quad & BW_{Target}(i) \geq \sum_{j=0,j\neq i}^{N-1} (BW(j) \times \frac{W_{SL_{out}(j),TL_{out}(j)}}{W_{SL_{in}(j),TL_{in}(j)}}) \\
& |W_{SL_{out}(k),TL_{out}(k)} - W_{SL_{out}(l),TL_{out}(l)}| \leq T_W, k, l = 0, \ldots, N-1, k \neq l. \\
& -M(j) \times TL_{out}(j) - (2 - M(j)) \times SL_{out}(j) < -M(j)(M(j) - 2), \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad j = 0, \ldots, N-1. \\
& SL_{out}(j) \in \{0, 1\} \text{ and } TL_{out}(j) \in \{0, 1, 2\} \text{ and } M(j) \in \{0, 1, 2\}
\end{aligned}
\tag{4.3}
$$

where $BW(j)$ denotes bandwidth of video incoming from client $j$, $SL_{in}(j)$ and $TL_{in}(j)$ denote the spatial and temporal layer of the incoming video of the client $j$, $SL_{out}(j)$ and $TL_{out}(j)$ denote the spatial and temporal layer of the outgoing video of the client $j$, N denotes number of clients in the video-conference, $BW_{Target}(i)$ denotes the estimated available bandwidth for client $i$, $TW$ denotes the threshold for weight difference, $W_{SL_{out}(j),TL_{out}(j)}$ denotes approximated ratio and $M(j)$ denotes the motion level of the client $j$. $M(j)$ is equal to 0 for low motion, 1 for moderate motion and 2 for high motion.

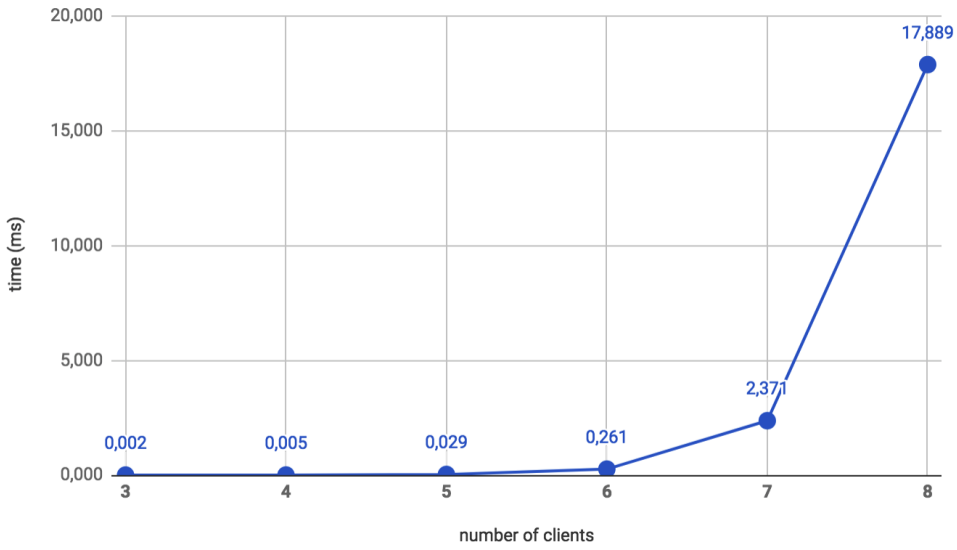


**Figure 4.3:** Execution time of the layer selection algorithm at the SFU with respect to the number of clients.

### *4.3.3 Solution of the Optimization Problem*

This optimization problem needs to be solved for every client $i$ for $i = 0, \ldots, N-1$, separately. It is solved by exhaustive search method using Algorithm 5. Although exhaustive search is NP-complete, a real-time solution is still possible since the number of client is typically small. Figure 4.3 shows the execution time of the layer selection algorithm at the SFU with respect to the number of clients.

The inputs to Algorithm 5 are the number of clients $N$, the bitrate $BW_{in}$ of all incoming video streams, the motion activity measure $Motion$ for each incoming video, the coefficient vector $Weights$ indicating the ratio of bitrate of each combination of layers to that of full video, a threshold $T_W$ to evaluate the fairness of number of streams from each client, the bandwidth $BW_{target}$ of the target client. The Algorithm finds the optimal number of spatial and temporal layers, $SL_{out}$ and $TL_{out}$, for all incoming video streams to be forwarded to a particular client and repeat it for all clients. The outputs are $N \times N$ matrices, e.g., the output entry $SL_{out}(j, i)$ is the number of spatial layers from client $j$ forwarded by the SFU to client $i$.

$W$ is the matrix of weights that are used to calculate objective function $\Delta$ and

**Algorithm 5:** Layer Selection at the SFU

**Input:** $N$, $BW_{in}$, $Motion$, $Weights$, $T_W$, $BW_{target}$
**Output:** $SL_{out}$ , $TL_{out}$
**foreach** *Picture Group* **do**
    Initialize $A$ (array of size $N$) to zero
    Initialize $SL_{out}$ and $TL_{out}$ ($N \times N$ arrays) to zero
    $min = inf$
    **for** $j = 0 : N - 1$ **do**
        W(:, j) = $Weights$
        **if** *( $Motion(j)$ = high )* **then**
            $W(0, j) = inf$
            $W(3, j) = inf$
        **else if** *( $Motion(j)$ = low )* **then**
            $W(0, j) = inf$
            $W(1, j) = inf$
            $W(2, j) = inf$
    **for** $i = 0 : N - 1$ **do**
        **for** $count = 0 : 6^{N-1} - 1$ **do**
            $A \leftarrow$ convert *count* to senary (base-6) numeral system
            $A(j), j = 0, ..., N - 2 \leftarrow$ digits of $A$
            **for** $j = 0 : N - 2$ **do**
                $W_{selected}(j, count) = W(A(j), j)$
            $\Delta = \Delta(i, count)$ given by Equation 4.2.
            **if** *( $\Delta \geq 0$) and ($\Delta \leq min$) and (max($W_{selected}(:, count)$) - min($W_{selected}(:, count)$)) $\leq T_W$ )* **then**
                $min = \Delta$
                **for** $j = 0 : N - 2$ **do**
                    $SL_{out}(j, i), TL_{out}(j, i) \leftarrow A(j)$ according to Table 4.2

**Table 4.2:** Mapping between each digit of A and combination of spatial and temporal layers

| **A** | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| **SL** | 0 | 0 | 0 | 1 | 1 | 1 |
| **TL** | 0 | 1 | 2 | 0 | 1 | 2 |

initialized to the input $Weights$, array of the all possible ratios for each client. For a video with high motion content, the possibilities with the lowest temporal layer and for a video with low motion content, the possibilities with the lowest spatial layer are not calculated while looking for the optimal solution. This constraint is applied to the algorithm by eliminating weights with undesirable layers by setting them to infinity. According to the amount of the motion in a particular video, corresponding possible weights are eliminated for this client between lines 9-15. This process is repeated for each client between lines 7-15.

For $N$ clients, 2 spatial and 3 temporal layers, the optimal spatial and temporal layers to forward to client $i$ that minimize the objective function $\Delta$ are selected among $6^{N-1}$ combinations between lines 16-26. The $6^{N-1}$ possible combinations are enumerated by the variable $count = 0, ..., 6^{N-1} - 1$. When $count$ is converted to senary (base 6) numeral system, the digits of the senary number $A$ determine which weight is used for each client to calculate objective function.

The calculated objective function needs to be positive to avoid the situation that the total bitrate of the video streams exceed the estimated bandwidth. Also, the difference between weights needs to be below the threshold $T_W$ for fair degradation. Among the combinations of the layers that provide this requirements, the one that minimizes the $\Delta$ is selected as an optimal solution. The optimal selection $A$ can be mapped to corresponding spatial and temporal layer for each incoming video stream according to Table 4.2.

## 4.4 Experimental Results

### 4.4.1 Experimental Test-bed

Experiments were conducted over a local area network with three computers connected to each other by a switch. Both computers run WebRTC client on Chrome web browser and one of them runs a signaling server and SFU. Clients run at different computers because VP9-SVC encoding is computationally expensive. With this architecture, the risk of CPU overuse which affects performance and analysis negatively is decreased.

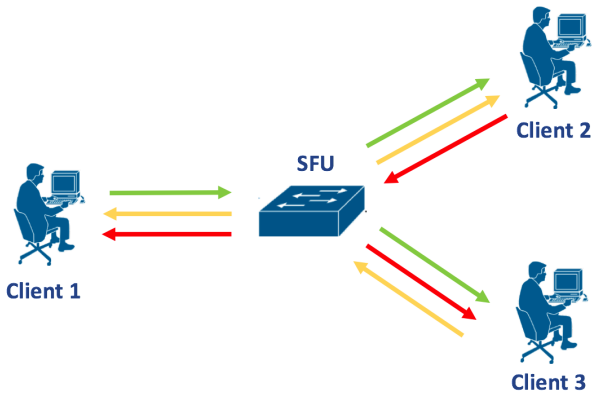


**Figure 4.4:** Test environment of multi-party RTC.

VP9-SVC functionality only works if Chrome with the most recent version used and started with the following flag :

```
--force-fieldtrials=WebRTC-SupportVP9SVC/EnabledByFlag_2SL3TL/
```

To conduct consistent and comparable experiments, for all tests the same raw video with 640x480 resolution and 30 fps is used. Y4M test file is fed to getUserMedia() instead of live camera input with the following flag :

```
--use-fake-device-for-media-stream --use-file-for-fake-video-capture=path/to/file.y4m
```

For the multi-party RTC experiments and analysis, the Janus gateway is used. Janus is chosen instead of other SFU solution alternatives because of its highly flexible architecture, efficient utilization of CPU and memory consumption in comparison with

others[22].

Janus is an open-source WebRTC Gateway allowing WebRTC clients to interact with legacy real-time communication technologies[23]. Janus has a modular architecture with a core and pluggable modules. Figure 4.5 shows the overview of the architecture. Any specific feature/application is provided by server side plugins, while the core handles the high level communication like WebRTC-related protocols.

VP9-SVC Video Room plugin is used for implementing this videoconferencing application. VP9-SVC Video Room plugin is based on Video Room plugin, but it has a key difference: providing support to the VP9 SVC layer selection.

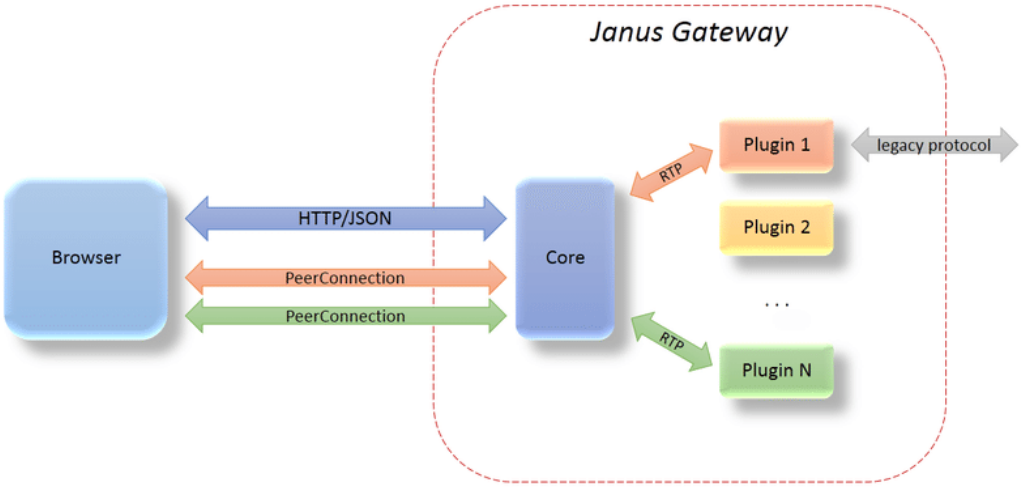


**Figure 4.5:** Janus modular architecture[22].

In order to emulate cross network traffic, we limit available uplink capacity of computers using the software [20]. 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 at both clients and SFU.

A Javascript library named getStats.js [24] is used to obtain the necessary current call statistics as drafted by W3C [25] like received video rates, sent video rates, number of frames decoded, video resolution, etc.

### 4.4.2 Results

We compare the performances of our proposed rate control with using motion and rate control without motion for multi-party WebRTC videoconferencing. We conducted experiments under identical network conditions with three clients and an SFU. When running the scalable VP9 encoder with three temporal and two spatial layers, the maximum video coding rate (with all layers) is set to 1024 kbps in the CBR mode. All clients perform scalable VP9 coding where the default spatial resolution of the base layer is 320x240 and enhancement layer is 640x480, and three temporal layers for 7.5 fps, 15 fps and 30 fps. But resolutions of the spatial layers can change because of the adaptation at the clients under congestion.

We present the video bitrate and frame rate for all links. All experiments lasted 250 sec, where there was high subject motion at the video stream of Client 1 and low motion at Client 2. We limited the available network bitrate of the link between SFU and Client 3 to 1300 kbps at 40 seconds to compare the response of SFU with different rate control models to this limited available bitrate and the available network bitrate of the link between Client 1 and SFU to 600 kbps at 27 seconds to compare the response of clients with different rate control models to this limited available bitrate.

*Rate Control Using Motion*

*Streaming from Client 1 to SFU:* The video of Client 1 has high motion during the whole experiment. Between 100 and 250 seconds, the available bandwidth is reduced to 600 kbps. In response to the congestion, the motion-based adaptation manager at Client 1 decreases the spatial resolution of the base layer to 240x180 and the enhancement layer to 480x360 while preserving temporal layer at the highest because of the high motion in the video.

*Streaming from Client 2 to SFU:* The video of Client 2 has very limited motion and there is no congestion at the bandwidth between Client 2 and SFU during the whole experiment. Video stream can be sent at the maximum encoding bitrate, 1024

kbps. Resolution is set to 640x480 and frame rate is around 30 fps during 250 seconds.

*Streaming from SFU to Client 3:* The link between SFU and Client 3 is not congested initially. Between 40 and 250 seconds, the available bandwidth is reduced to 1300 kbps. The total bitrate received by SFU is around 2048 kbps up to 40th second. SFU drops one spatial layer from the video stream of the Client 1 and one temporal layer from the video stream of the Client 2 between 40 and 100 seconds, in response to the congestion. After 100 seconds, the total bitrate of the incoming streams decreases because of the congestion occurred between Client 1 and SFU. SFU starts to send full spatial layers of the Client 1's video stream and drops one more temporal layer from the Client 2. By this way, SFU maximizes the usage of the available bandwidth with a fair degradation.

The results of the experiments are shown in Fig. 4.6, Fig. 4.7, Fig. 4.8, Fig. 4.9, Fig. 4.10, Fig. 4.11, and Fig. 4.12. Under congestion, available bandwidth usage of the link between Client 1 and SFU is % 101.29, and between SFU and Client 3 is % 86.97. Table 4.3 shows the average bitrate, frame rate and resolution for all links. We can observe that the average frame rate of the video 1 with high motion is sufficiently high when spatial resolution is reduced. On the other hand, spatial resolution of the video 2 with low motion is preserved during experiment. The bitrates of the video stream 1 and 2 that sent from SFU to Client 3 are close and they split the available bandwidth between SFU and Client 3 fairly.

**Table 4.3:** Average results for video stream of Client#1 and Client#2, received by Client#3

| | **Client1 to SFU** | **Client2 to SFU** | **SFU to Client3 (video stream1)** | **SFU to Client3 (video stream2)** |
|---|---|---|---|---|
| **Bitrate** | 782.8 | 1041.6 | 606.4 | 661.4 |
| **Frame rate** | 29.9 | 29.9 | 28.9 | 12.6 |
| **Resolution** | 543x407 | 640x480 | 464x348 | 640x480 |

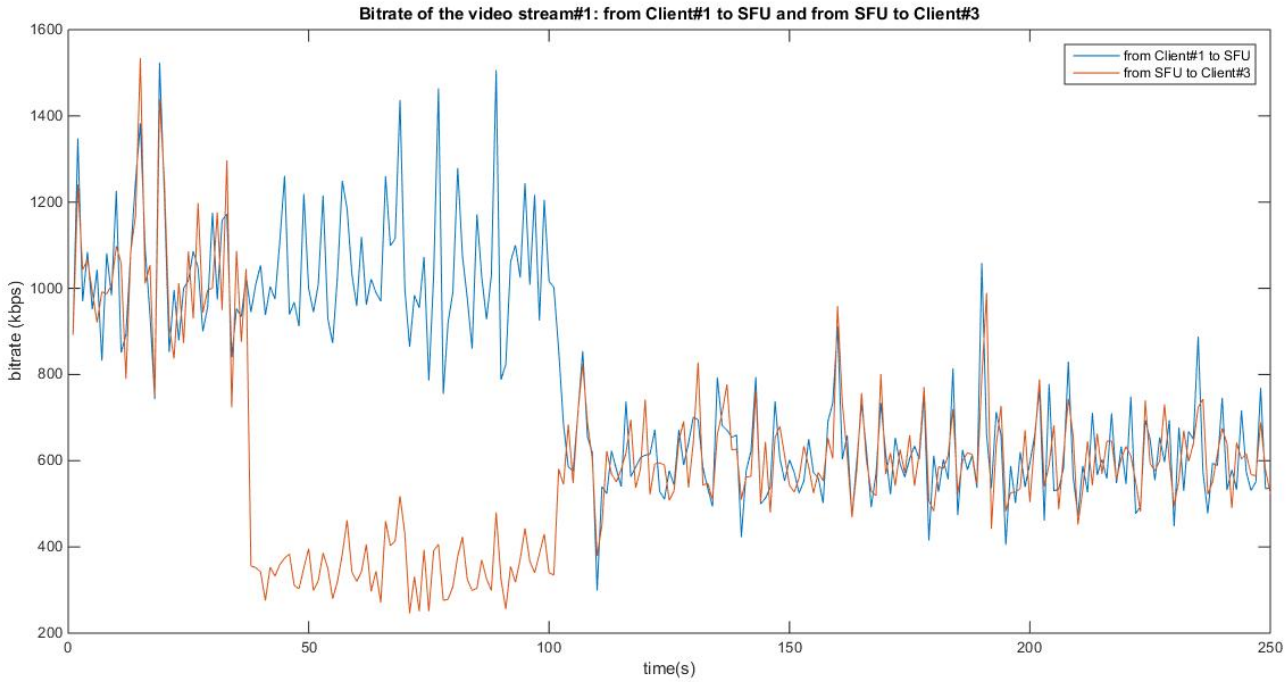


**Figure 4.6:** Bitrate of the video stream#1: from Client#1 to SFU and from SFU to Client#3

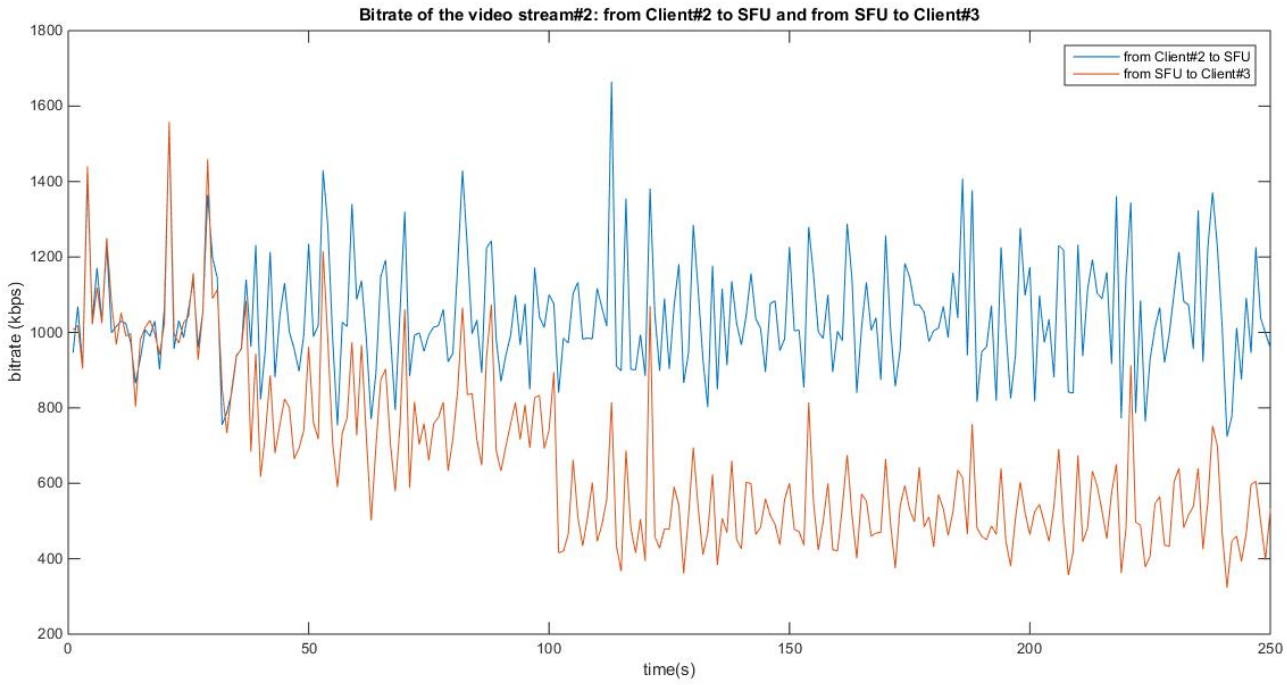


**Figure 4.7:** Bitrate of the video stream#2: from Client#2 to SFU and from SFU to Client#3

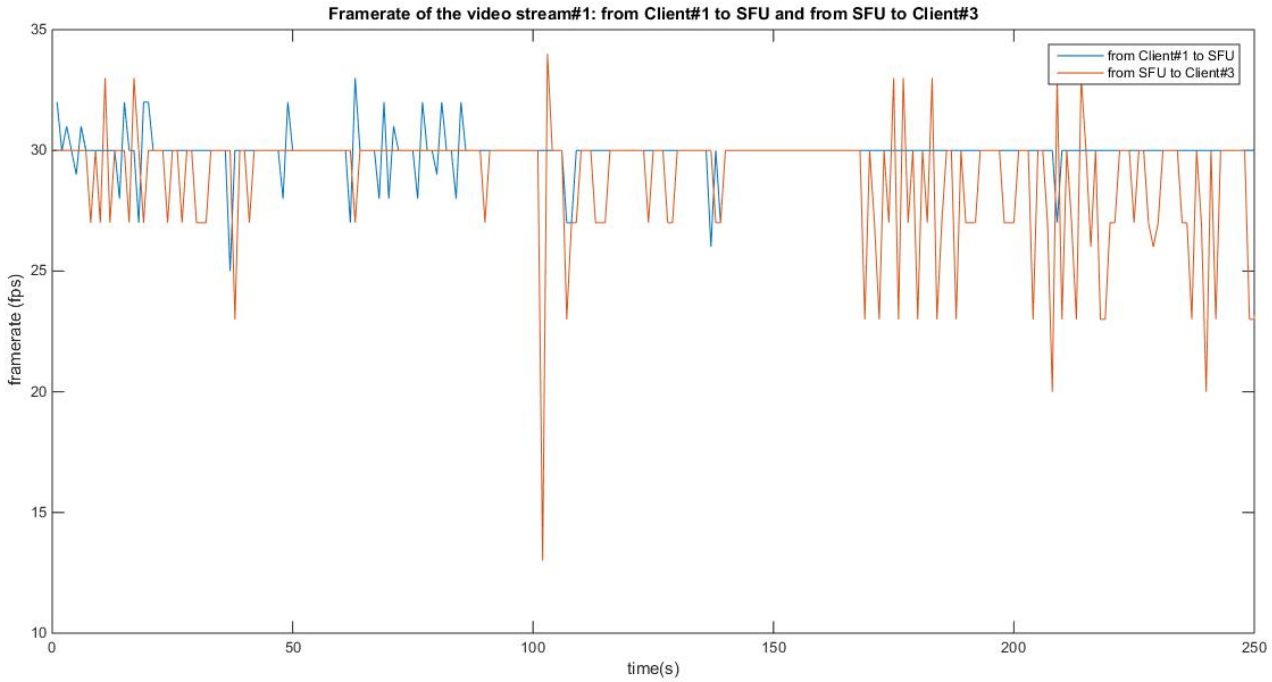


**Figure 4.8:** Frame rate of the video stream#1: from Client#1 to SFU and from SFU to Client#3

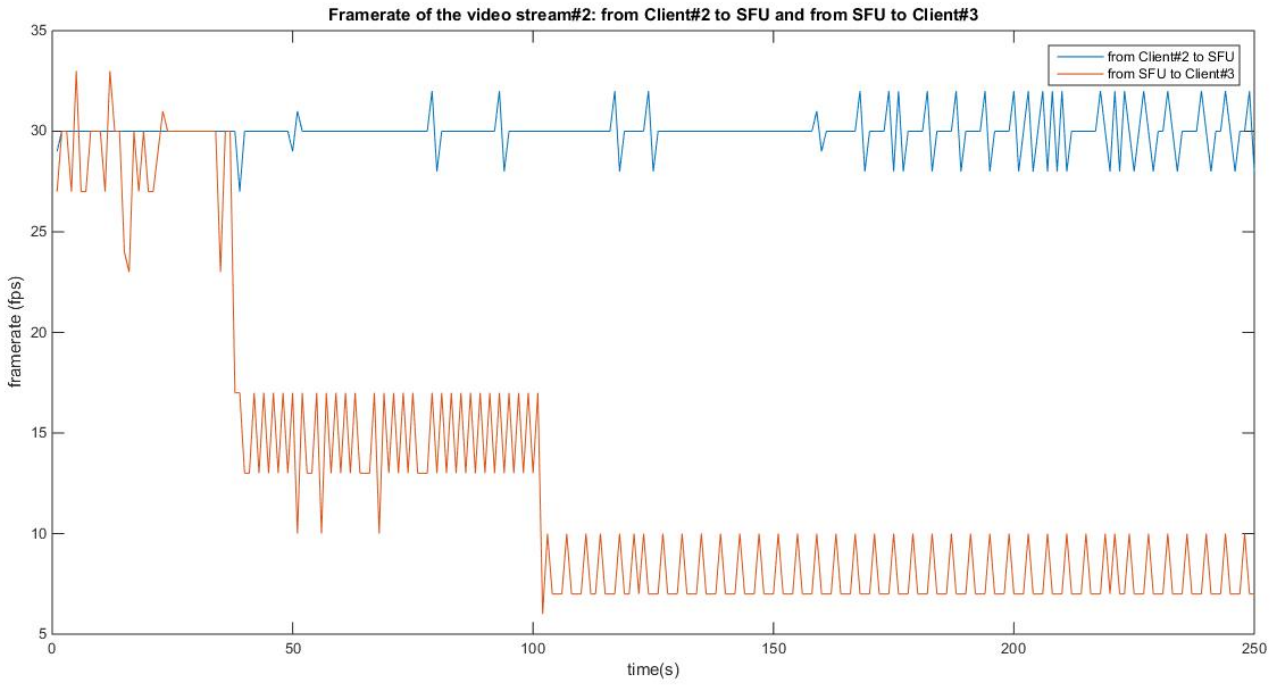


**Figure 4.9:** Frame rate of the video stream#2: from Client#2 to SFU and from SFU to Client#3

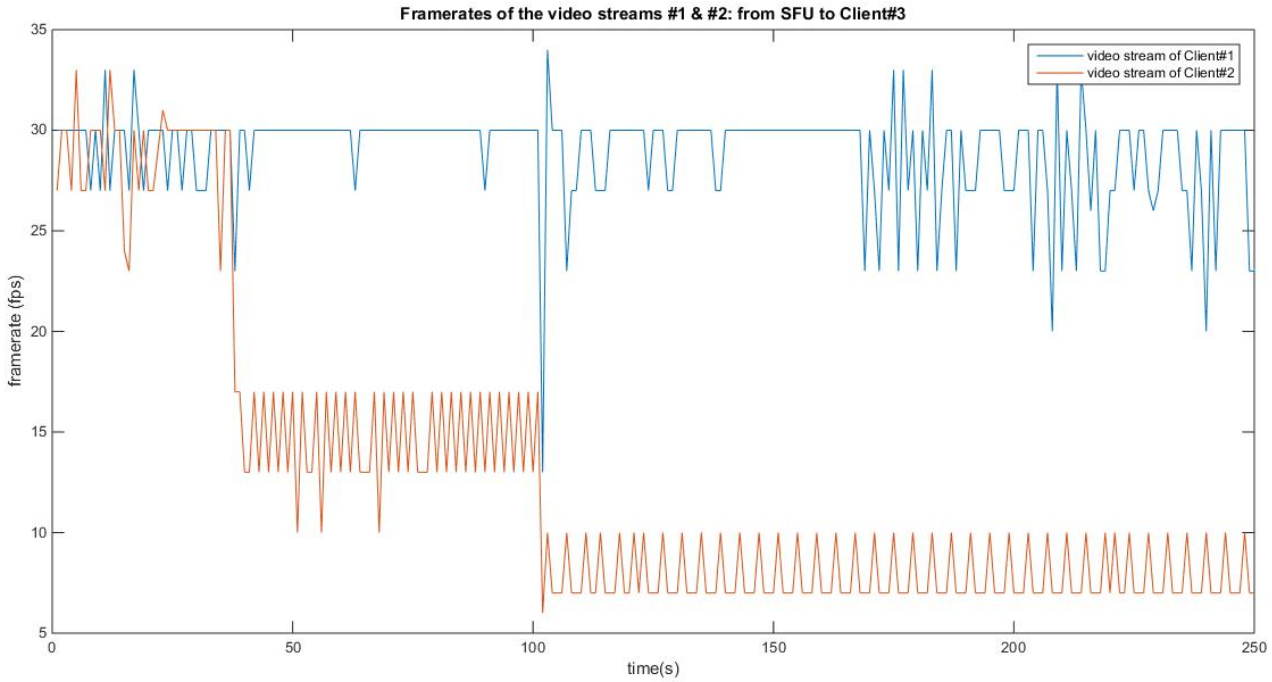


**Figure 4.10:** Frame rates of the video streams #1 & #2: from SFU to Client#3

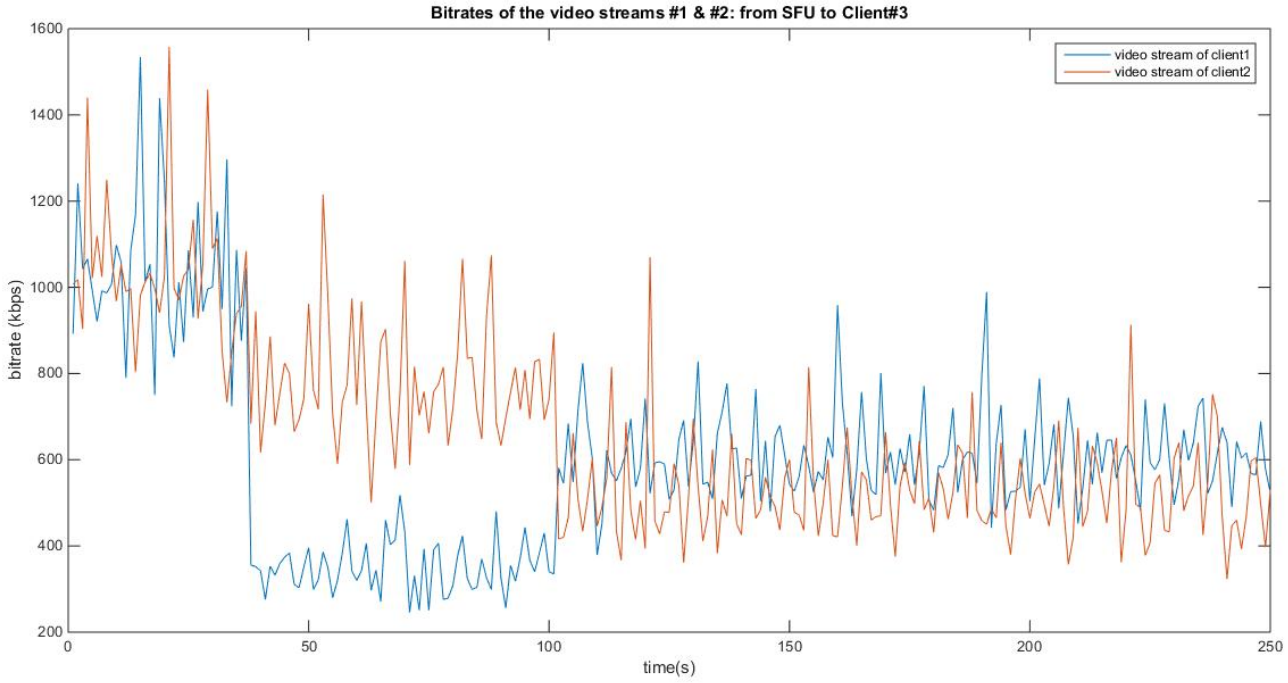


**Figure 4.11:** Bitrates of the video streams #1 & #2: from SFU to Client#3

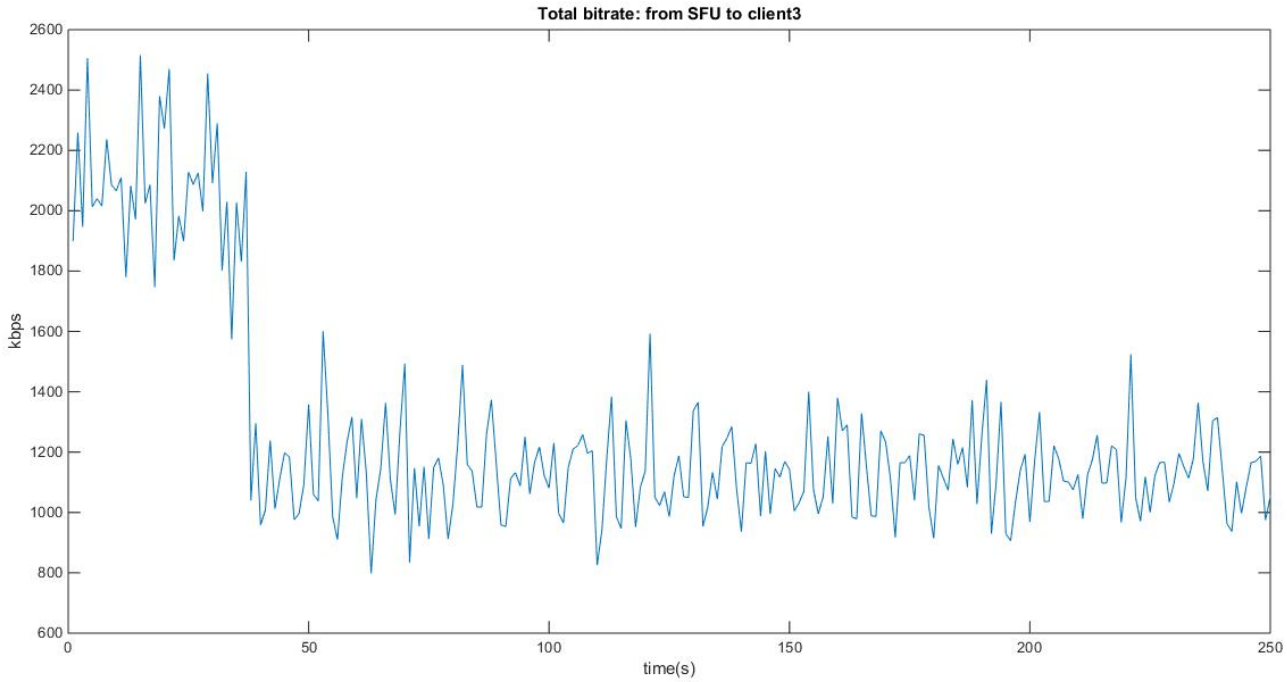


**Figure 4.12:** Total bitrate of the video streams #1 & #2: from SFU to Client#3

*Rate Control Without Using Motion*

For layer selection at SFU, the optimization problem that is introduced in Section 4.3 is solved without considering fairness between degradation of layers from different videos or motion amount in videos. Rate is controlled with the aim of only maximizing the bandwidth usage. For the rate control at client, the default control mechanism of the browser is used.

*Streaming from Client 1 to SFU:* The video of Client 1 has high motion during the whole experiments. Between 100 and 250 seconds, the available bandwidth is reduced to 600 kbps. In response to the congestion, Client 1 decreases the video quality (increases quantization parameter) while preserving resolution at the 640x480 and temporal layer at the highest.

*Streaming from Client 2 to SFU:* The video of Client 2 has very limited motion and there is no congestion at the bandwidth between Client 2 and SFU during the whole experiment. Video stream can be sent at the maximum encoding bitrate, 1024 kbps. Resolution is set to 640x480 and frame rate is around 30fps during 250 seconds.

*Streaming from SFU to Client 3:* The link between SFU and Client 3 is not congested initially. Between 40 and 250 seconds, the available bandwidth is reduced to 1300 kbps. The total bitrate received by SFU is around 2048 kbps up to 40th second. SFU drops two temporal layers from the video stream of the Client 1 and one temporal layer from the video stream of the Client 2 between 40 and 100 seconds, in response to the congestion. Despite the fact that video of the Client 1 contains high motion activity, the frame rate of the video decreases up to 7 fps between 40 and 100 seconds. After 100 seconds, the total bitrate of the incoming streams decreases because of the congestion occurred between Client 1 and SFU. SFU starts to send 1 spatial layer and 3 temporal layers of the Client 1's video stream and full spatial and temporal layers of the Client 2's video stream. SFU maximizes the usage of the available bandwidth but the layers of videos are not degraded fairly.

The results of the experiments are shown in Fig. 4.13, Fig. 4.14, Fig. 4.15, Fig. 4.16, Fig. 4.17, Fig. 4.18, and Fig. 4.19. Under congestion, the available bandwidth usage of the link between Client 1 and SFU is % 102.04, and between SFU and Client 3 is % 97.67. Despite the high usage of the available bandwidth, the rate control without considering motion causes motion jitter and blocking artifacts. This implementation performs poorly in terms of subjective video quality.

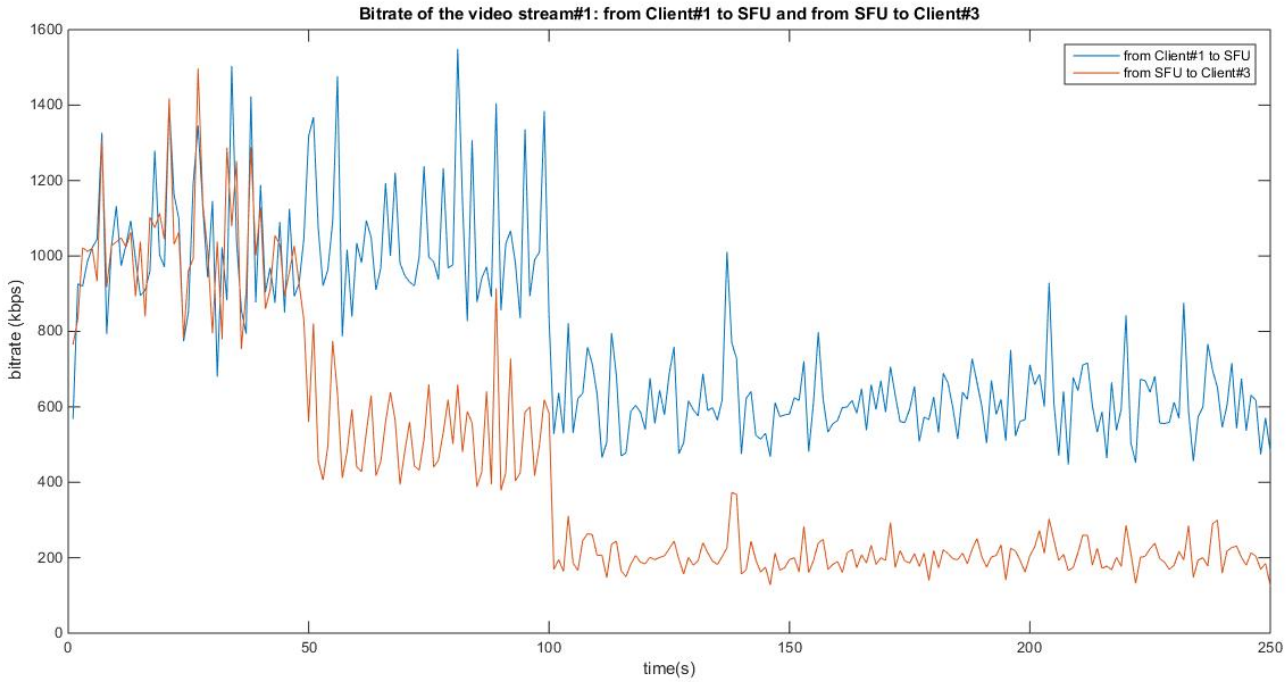


**Figure 4.13:** Bitrate of the video stream#1: from Client#1 to SFU and from SFU to Client#3

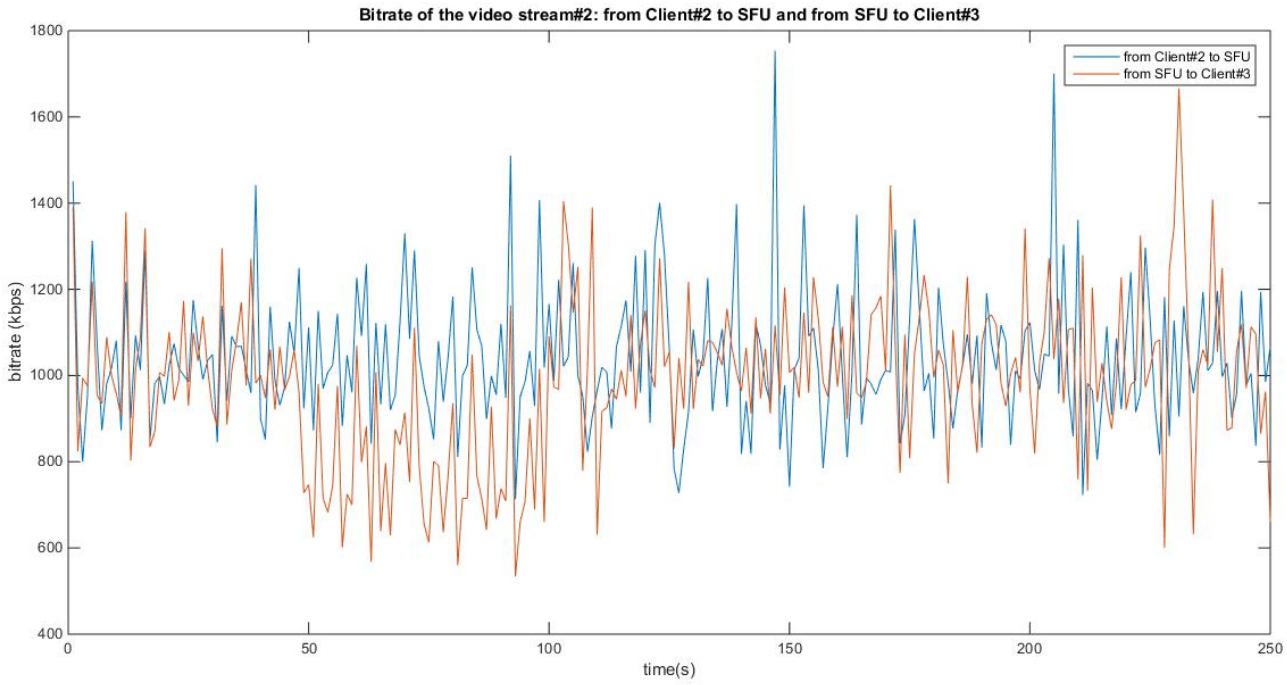


**Figure 4.14:** Bitrate of the video stream#2: from Client#2 to SFU and from SFU to Client#3

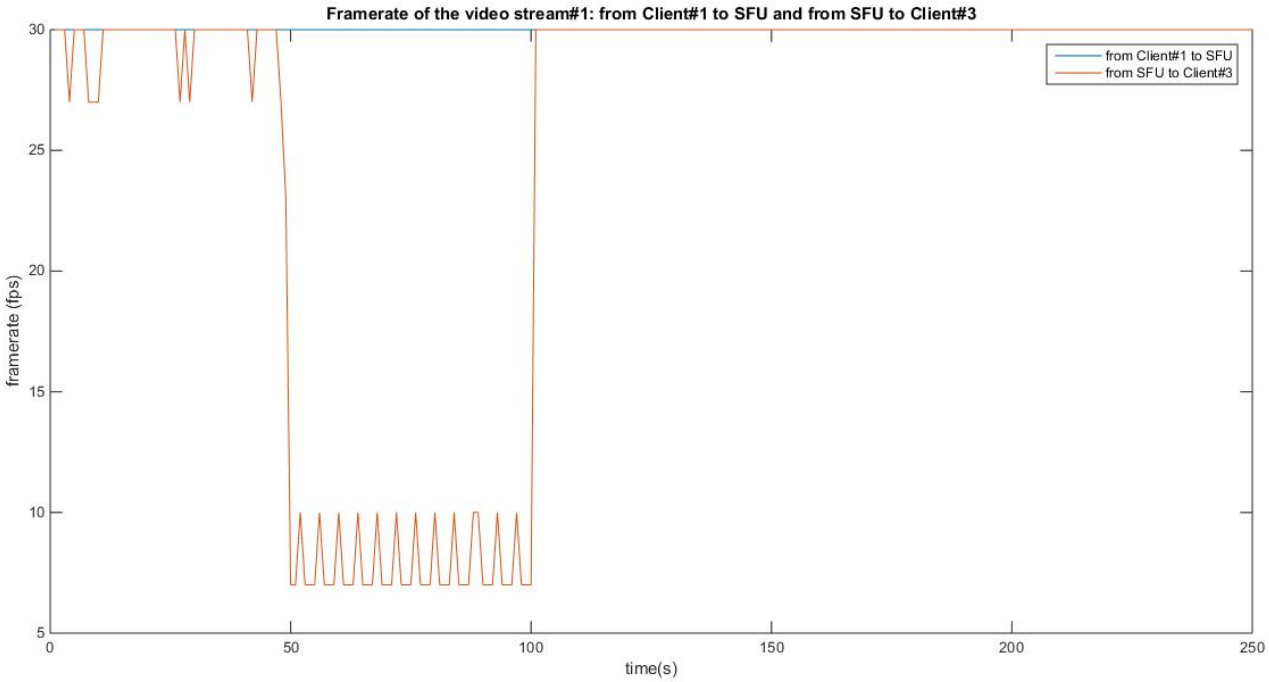


**Figure 4.15:** Frame rate of the video stream#1: from Client#1 to SFU and from SFU to Client#3

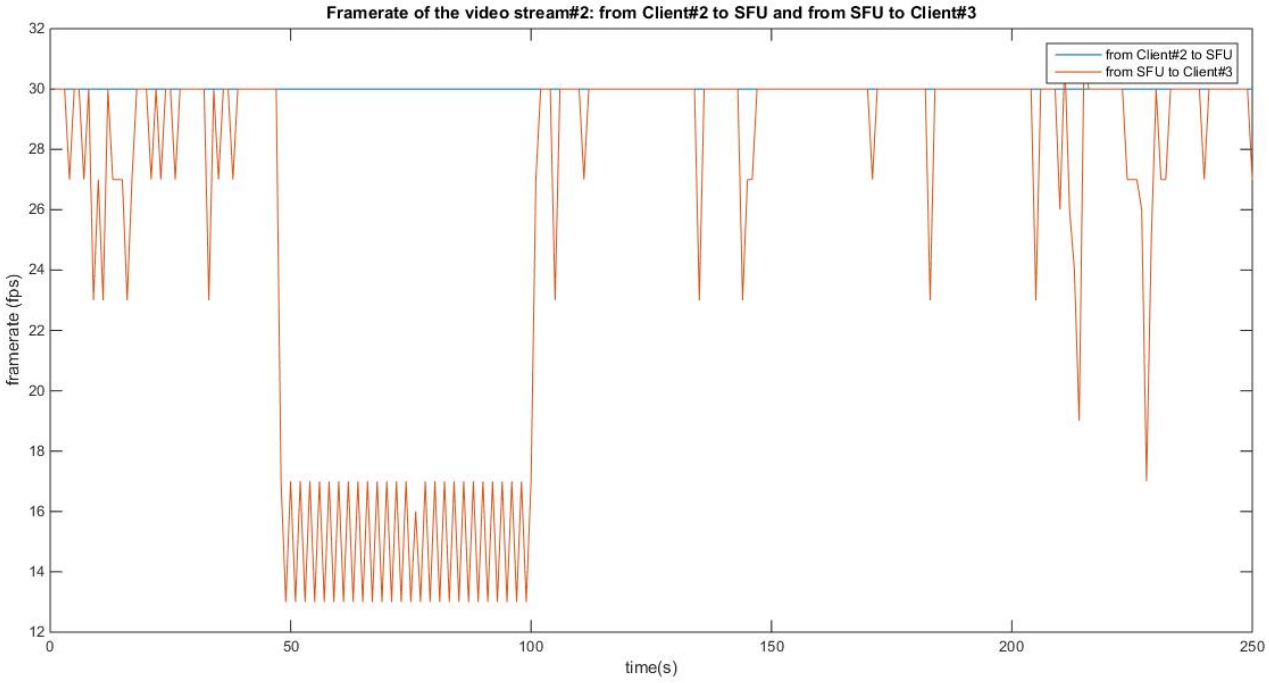


**Figure 4.16:** Frame rate of the video stream#2: from Client#2 to SFU and from SFU to Client#3

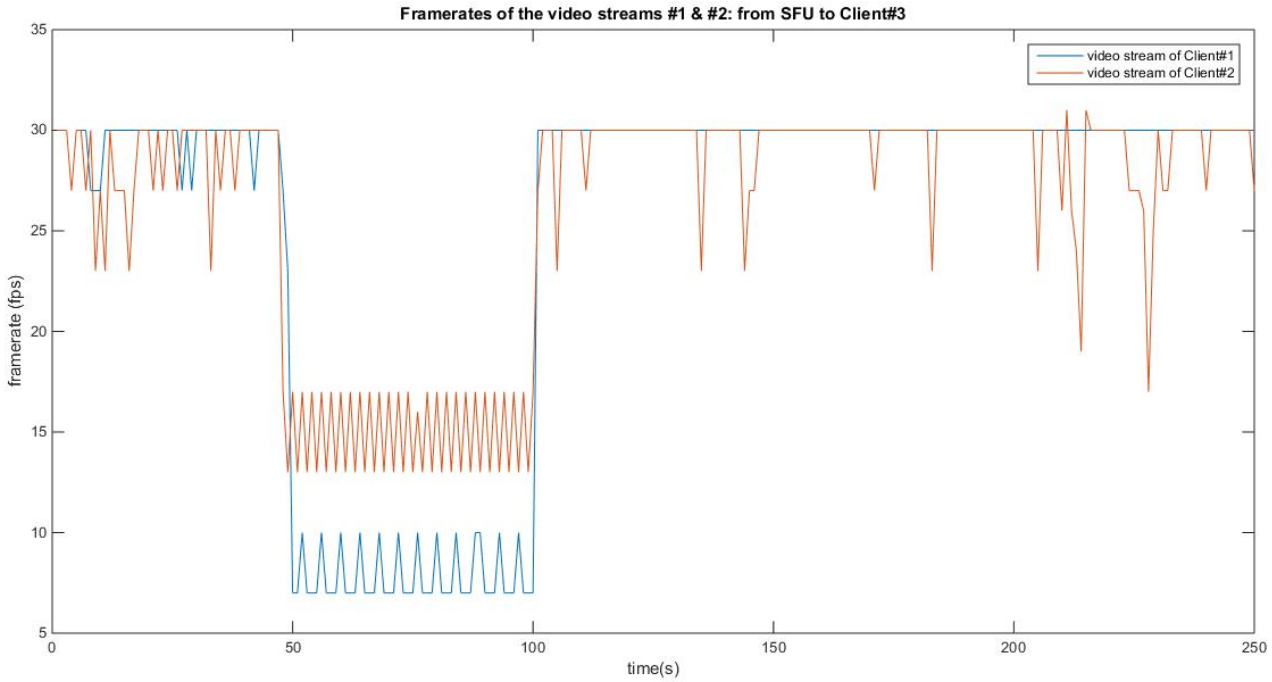


**Figure 4.17:** Frame rates of the video streams #1 & #2: from SFU to Client#3

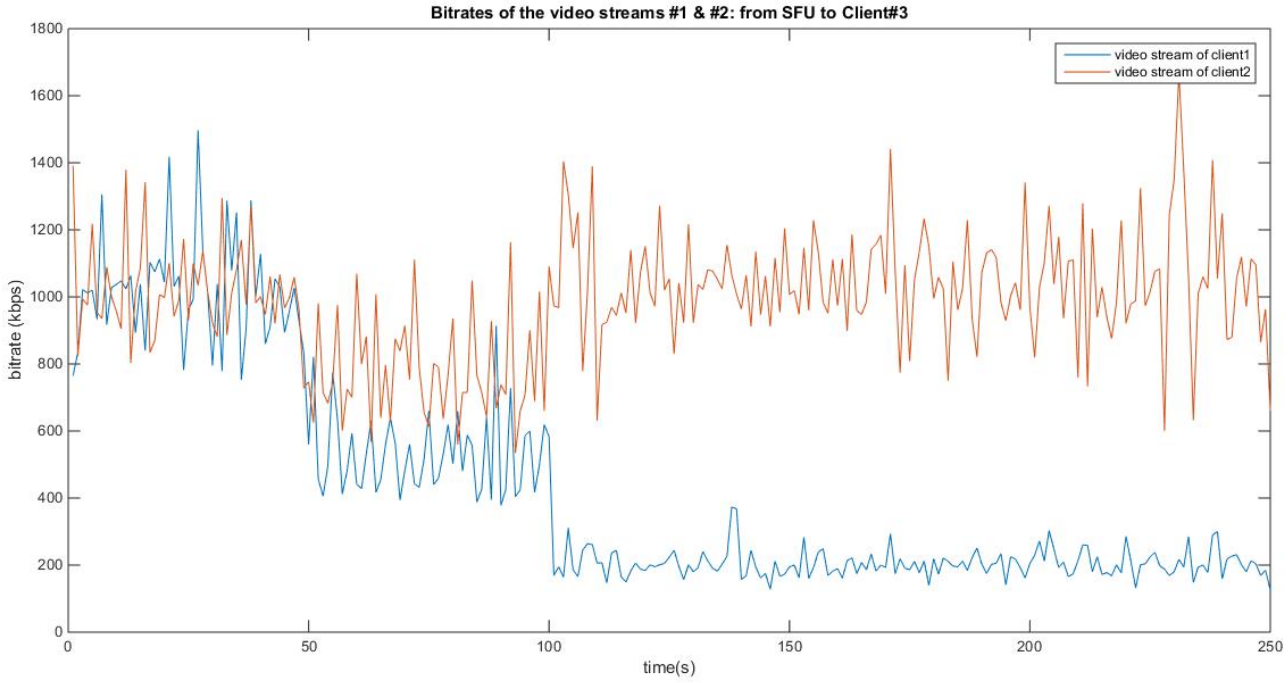


**Figure 4.18:** Bitrates of the video streams #1 & #2: from SFU to Client#3

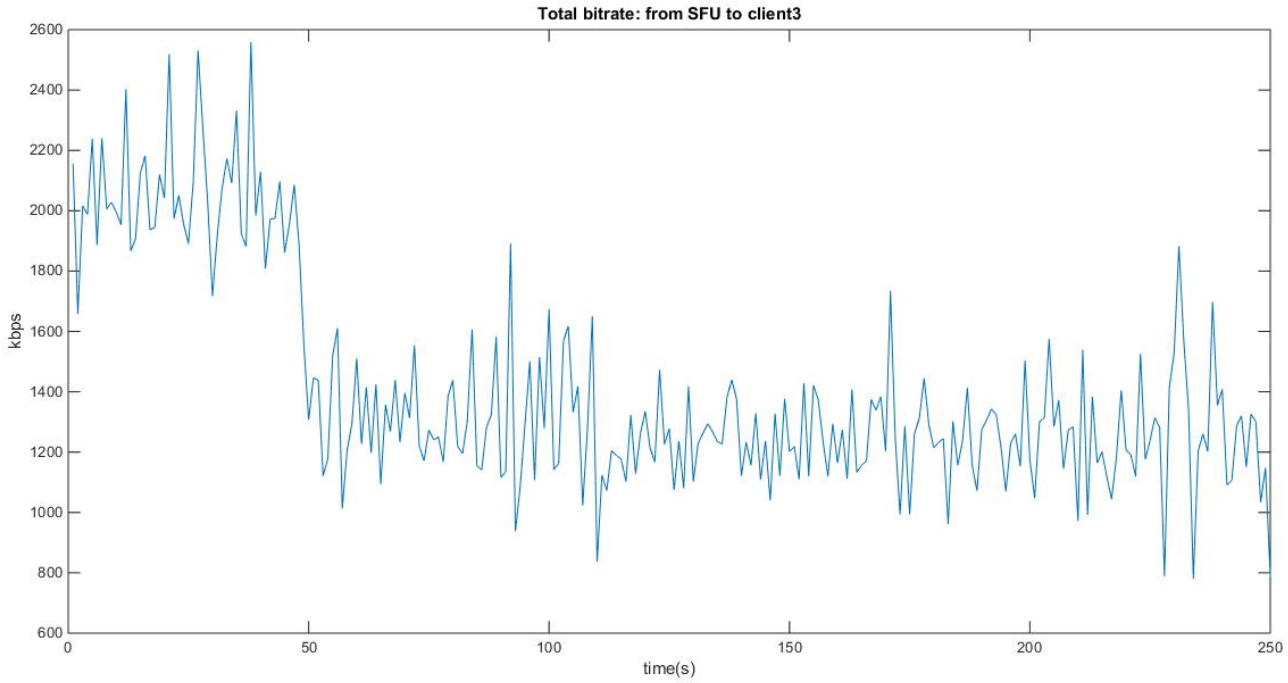


**Figure 4.19:** Total bitrate of the video streams #1 & #2: from SFU to Client#3

# Chapter 5

# CONCLUSION

This thesis introduces motion-based management of the layer selection of scalable coded video for both point-to-point WebRTC sessions and multi-party WebRTC videoconferencing. The default WebRTC CBR control (non-scalable encoder) increases the quantization parameter and/or reduce frame rate to maintain the desired source bitrate. Rate control implementations that do not consider the content of the video are not sufficient to achieve desirable subjective video quality. 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 trade-off between coarser quantization, which results in blocking artifacts and spatial resolution reduction, which results in some blurring.

In this thesis, we introduced motion-based rate control for point-to-point RTC and evaluate the performance and video quality of our proposed algorithm and compared to default single-layer rate control. Secondly, we introduced motion-based rate control for both SFU and client on the purpose of multi-party videoconferencing. We compared the performance of our proposed rate control with default browser rate control at client and rate control without considering motion at SFU.

Our results indicate that the proposed motion-based spatial-resolution adaptive rate control model achieves sufficiently high frame rates and reasonable quantization parameter values for video with high motion and high resolution and reasonable quantization parameter values for video with low motion which yields more pleasing video quality. Mixed spatio-temporal scalable coding together with our proposed motion-based rate control algorithm can significantly improve visual video quality under network congestion for both point-to-point and multi-party RTC.

# BIBLIOGRAPHY


[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. 826–839, 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. 1103–1120, 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] R. Civanlar, A. Eleftheriadis, O. Shapiro, “System and Method for a Conference Server Architecture for Low Delay and Distributed Conferencing Applications,” U.S. Patent No. 7,593,032, Sept. 2009.

[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. 339–343, 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] F. L. van Nes, J. J. Koenderink, H. Nas, and M. A. Bouman, “Spatiotemporal modulation transfer in the human eye,” Jour. of the Optical Society of America, vol. 57, no. 9, Sept. 1967.

[15] G. Bakar, R. A. Kirmizioglu, and A. M. Tekalp, “Motion-based adaptive streaming in WebRTC using spatio-temporal scalable VP9 video coding," IEEE Globecom, Singapore, Dec. 2017.

[16] WebRTC source code, https://chromium.googlesource.com/external/webrtc/

[17] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," RFC 3550, July 2003

[18] S. Holmer, H. Lundin, G. Carlucci, L. De Cicco, S. Mascolo, "A Google Congestion Control Algorithm for Real-Time Communication," July 2016

[19] M. Westerlund, S. Wenger, "RTP Topologies," RFC 7667, November 2015

[20] NetLimiter, available online "https://www.netlimiter.com/"

[21] S. G. Murillo, G. Garcia, "Chrome's WebRTC VP9 SVC Layer Cake", February, 2017

[22] Amirante, A. and Castaldi, T. and Miniero, L. and Romano, S. P., "Janus: A General Purpose WebRTC Gateway," Proc. of the Conf. on Principles, Systems and Applications of IP Telecommunications, pp. 7:1–7:8, 2014

[23] Janus Gateway, https://github.com/meetecho/janus-gateway

[24] getStats JavaScript library, https://github.com/muaz-khan/getStats

[25] W3C, "Identifiers for WebRTC's Statistics API," May 2016. [Online]. Available: http://www.w3.org/TR/webrtc-stats/

# VITA

GONCA BAKAR was born in İzmir, Turkey on 18th September. She received her B.S. degree in Electrical and Electronics Engineering from Boğaziçi University, Istanbul, Turkey in 2015. She joined the M.S. program in Electrical and Electronics Engineering at Koc University in 2015 as a research and teaching assistant. During her graduate studies she worked on adaptive video streaming, scalable video encoding, and WebRTC. She has co-authored a conference paper that was presented in the IEEE Global Communications Conference (Globecom) in December 2017.