Improving the Efficiency of WebRTC Layered Simulcast Using Software Defined Networking

Agnieszka Chodorek[1], Robert R. Chodorek[2(✉)], and Krzysztof Wajda[2]

[1] Kielce University of Technology, Al. 1000-lecia P.P. 7, 25-314 Kielce, Poland
a.chodorek@tu.kielce.pl
[2] The AGH University of Science and Technology, Al. Mickiewicza 30, 30-059 Krakow, Poland
{chodorek,krzysztof.wajda}@agh.edu.pl
https://telekomunikacja.agh.edu.pl/

Abstract. This study proposes a conference bridge that cooperates with both the WebRTC layered simulcast and the Software Defined Networking architecture in order to improve WebRTC video streaming. The proposed bridge divides the functionality of a classic Selective Forwarding Unit into two parts. The selection of layers is performed by the SDN controller and the forwarding of layered video is still accomplished by the bridge. The bridge and the SDN controller exchange data on the state of the transmitted video stream and the state of the network. The proposed solution was implemented in the Jitsi Videobridge and tested in the GEANT testbed network. The results showed that our solution significantly reduces problems related to available throughput overshooting, which is typical for layered simulcast.

Keywords: WebRTC · Software Defined Networking · OpenFlow system · Quality of Service

1 Introduction

Forced by the events of the last few years, the rapid development of telemedicine, e-learning and all kinds of telework has become both an opportunity and a challenge for multimedia communication technology. The emerging technology that was able to both seize the opportunity and meet the challenges is the Web Real-Time Communications (WebRTC) [1,2], which introduced native real-time communication to the web. It offers some interesting features supporting efficient multimedia communication, including layered simulcast.

This work was supported by the Polish Ministry of Science and Higher Education with the subvention funds of the Faculty of Computer Science, Electronics and Telecommunications of AGH University.

W. Wang and J. Wu (Eds.): BROADNETS 2023, LNICST 511, pp. 13–28, 2023.
https://doi.org/10.1007/978-3-031-40467-2_2

1.1 Simulcast

In the field of information and communication technologies (ICT), simulcast denotes simultaneous streaming of the same content, encoded to different formats, different bitrates, or (if layered codecs are used) to successive layers. So-created streams (layers) are directed to middleboxes that serve a recipient or group of recipients and intermediates in simulcast transmissions. Middleboxes are implemented as media servers or Selective Forwarding Units (SFUs).

Streams and sets of successive layers are selectively forwarded to the recipients, and selection is carried out by the congestion control mechanism. Depending on the implementation, the congestion controller can be placed in the receiver (receiver-driven simulcast) or in a middlebox (node-driven simulcast).

Let N be the total number of streams or layers. The controller dynamically chooses 1 of the N replicated streams (stream replication simulcast) or a set of M layers (layered simulcast) in order to achieve the best quality of transmitted media that the network is able to deliver. Layers are chosen successively, from 1 to M of N. The chosen stream or set of layers are re-sent by the SFU to the destination (see explanatory drawing showed in Fig. 1). The process of choosing the stream/layers is repeated according to changes to the network's state.

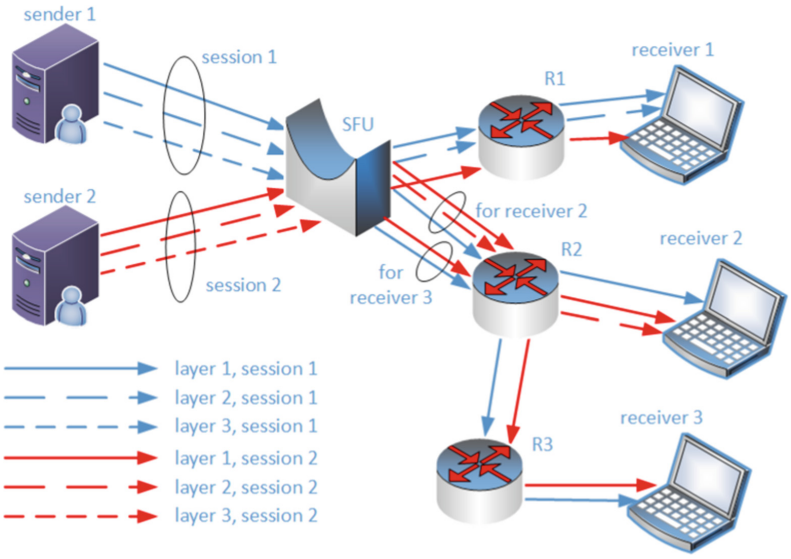

Fig. 1. The concept of layered simulcast. One SFU serves a group of recipients (receiver 1 to receiver 3).

1.2 WebRTC and WebRTC Simulcast

The WebRTC concept of native real-time communication in the Web was intended for peer-to-peer media communication between web browsers. However, the WebRTC-based conferencing system may be built as a centralized one,

with the use of middleboxes (working as conferencing bridges). The most popular open source middleboxes, used by the WebRTC, are the Jitsi Meet, the Edumeet, the Kurento and the Simple Realtime Server (SRS). The comparison of WebRTC open source SFUs used for video conferencing is presented in this paper [3]. WebRTC is able to provide a conferencing service, when the conference bridge is placed in a computing cloud [4,5].

Currently, WebRTC is implemented in all popular web browsers, which are run-time environments for WebRTC applications. WebRTC applications are written according to the Single-Page Application (SPA) paradigm, with the use of the HyperText Markup Language (HTML) version 5 (HTML5) and the JavaScript language. WebRTC sessions are established using the JavaScript Session Establishment Protocol (JSEP) and the Session Description Protocol (SDP). Media (audio and video) are transmitted using the Real-time Transport Protocol (RTP) operating on a top of the User Datagram Protocol (UDP). The WebRTC has sophisticated congestion control of media streams, composed of several internal mechanisms. They are implemented in Web browsers, e.g. TCP-friendly Rate Control (TFRC), the Google Congestion Control (GCC), and in middleboxes. One of the congestion control mechanisms implemented in the middleboxes is simulcast. Performance evaluation of the congestion control implemented in Web browsers is presented in [6,7], and implemented in middleboxes as presented in [8,9].

Simulcast is a relatively new feature of the WebRTC derived from multicasting [10]. Initially, WebRTC applications were able to use classic simulcast, based on stream replication. Since January 2019 [11], the first browser that supported WebRTC was equipped with Scalable Video Coding (SVC) versions of popular codecs (the VP8, VP9, and H.264 codecs). In 2019, the W3C Editor's Draft enabled the use of layered simulcast, but significant work is still ongoing and is documented in subsequent versions of this draft. The latest version is dated 13 February 2023 [12].

1.3 Motivations, Contributions and Organization of This Paper

In the paper [8] it was reported that the WebRTC's simulcasts (both the stream replication one and the layered one) were not able to perform precise and fast reaction to congestion, even in wireless local area networks, where propagation delays are small. Lack of precise information about current path states and the trial-and-error evaluation of the available throughput causes overshooting of the estimation of available bandwidth, which leads to high error rates.

The aim of this paper is to show that this adverse phenomenon may be eliminated (or, at least, limited) by using knowledge of the current network state. Such knowledge is available in the Software Defined Network (SDN) switches. There are examples of cooperation between SDN and WebRTC in the literature (such as the use of WebRTC signalling information to allocate resources in the SDN network [13], SDN-assisted IP-multicasting in the 5G network ensuring effective distribution of WebRTC transmissions to multiple recipients [14], or building a distributed SFU using SDN [15]), but so far SDN has not been used for simulcast management.

The main contributions of this paper are:

- The concept of SDN-assisted simulcast management, designed to work with SFU managing WebRTC simulcast sessions.
- Evaluation of the prototype implementation of the proposed system in the GEANT testbed.

The rest of this paper is organized as follows. In the next section the concept of the proposed simulcast management system is described. Experiments aimed at the evaluation of this concept in the GEANT testbed network are presented in the third section. The fourth section shows and discusses results of performed experiments. The last, fifth section, briefly concludes this paper.

2 Related Work

This section discusses three aspects of related work: WebRTC architecture standards, research on WebRTC simulcast, and research on SDN support for WebRTC.

2.1 WebRTC Architecture and Use Cases

The main achievements of IETF RTCWeb WG are defining milestones for WebRTC and are summarized in basic RFCs defining WebRTC use cases [16], used video codecs [17] and audio codecs [18]. The main use cases listed in [16] are: the simple video communication service, screen sharing, file exchange, multiparty video communication, online game, and video conferencing systems. Examples of WebRTC applications shown in the literature cover the secure delivery of multimedia content [19], the video communication service [20], video conferencing [21], video collaboration [21] and multi-party videoconferencing [13,15]. WebRTC was also used in the Web of Things (WoT) systems for medical applications [5] and in the WoT-based flying monitoring systems [22–24].

A brief overview of WebRTC architecture and main features is included in [25,26], then extended in [27] and [28]. The concept of using simulcast, with a description of the architecture, implementation scenarios and mapping on SDP and RTP settings is presented in [29].

2.2 WebRTC Simulcast

Loreto and Romano describe a state-of-the-art of WebRTC standard ecosystem [10], and one of the sections of their paper is devoted to simulcasting as a probable part of WebRTC-1.0. The authors indicate that simulcast issues are still under discussion.

Lin et al. in the paper [30] showed that although simulcast allows for both high scalability and the building of cost-effective solutions, there are situations where resource utilization is suboptimal. As a solution to this problem, the authors proposed a global stream orchestration (GSO) using a controller with the

full network information. With this knowledge, simulcast congestion control can omit sending simulcast streams that will not be received by any of the receivers.

Chodorek et al. in the paper [8] compare the adaptability and the congestion control ability of both variants of WebRTC simulcasting: stream replication simulcast and layered simulcast. Results showed that in a not-loaded (without background traffic) environment, where only static (infrastructural) limitations occur, both versions of simulcast give equally good QoS parameters. However, in the presence of competitive background traffic, layered simulcast gives greater throughput and lower error rates.

Grozev et al. in the paper [9] presented an experimental evaluation of the basic, stream replication simulcast, shown by the example of the teleconference, where each participant may send from one to three replicated streams. The stream replication simulcast was compared with a single-stream unicast. The authors also indicate the need of further research on WebRTC simulcasting.

Xhagjika et al. present in their work [31] a comparative study of WebRTC stream replication simulcast and unicast transmissions between end systems (WebRTC clients) and media servers located in computing clouds. Results show that simulcasting gives about a two times higher bit rate than congestion controlled (in a TCP-friendly manner) unicasting.

Bakar et al. in the paper [20] describes a mesh-connected WebRTC transmission system, in which a spatio-temporal layered simulcast is used. The original simulcast was supplemented by a proposed motion-adaptive layer selection algorithm. Results shows that the layered simulcast is beneficial for point-to-point mesh-connected WebRTC sessions. The authors in their next paper [32] extend their proposition to WebRTC sessions that use the SFU intermediate node.

Romano and Giangrande in the paper [33], and Romano et al. in the paper [19] describe an architecture envisaging a combination of multicast, simulcast and unicast communication scenarios in a hybrid, terrestrial-satellite network. Simulcasting may be used for adaptive streaming in a WebRTC-enabled access network.

2.3 SDN-Aided WebRTC

Real-time communication (RTC) video services require a network with personalized service delivery with a higher capacity and better QoE [34]. The SDN network allows for the cooperation with WebRTC to do that service.

Kirmizioglu et al. in the paper [13] describes a WebRTC videoconferencing system which uses scalable VP9 video. To provide QoS services WebRTC signalling goes to the Network Service Provider (NSP). The NSP operates over the SDN. All experiments are performed using Mininet.

Cox et al. in the paper [35] describe methods for detecting Rogue access points (RAPs) using the SDN network and the WebRTC architecture. All tests are performed using Mininet. Boubendir et al. in the paper [36] described SDN-enabled NFV for a Telco network. Analysis shows that network operators can offer special network processing and forwarding for WebRTC communication. In the paper [37] the same authors show that it is possible to create on-demand

services for a WebRTC-based application (WebRTC-based applications demand services of the TURN server from a network provider, and the network provider sets it up on demand).

In the paper [14] written by Kirmizioglu et al., the service manager which reserves bandwidth between mesh-connected WebRTC clients according to the rates agreed by them was proposed. This solution works over the SDN network.

3 The Concept of the SDN-Assisted Layered Simulcast

In the case of the classic WebRTC layered simulcast, the SFU can act as a relay, which selectively forwards appropriate layers to the destination. The idea of the proposed SDN-assisted simulcast is to divide the functionality of a classic SFU into two parts. The first one is related to the layers' forwarding and multiplication. It still remains under the responsibility of the SFU. The second one, related to the layers' selection, is transferred to the SDN.

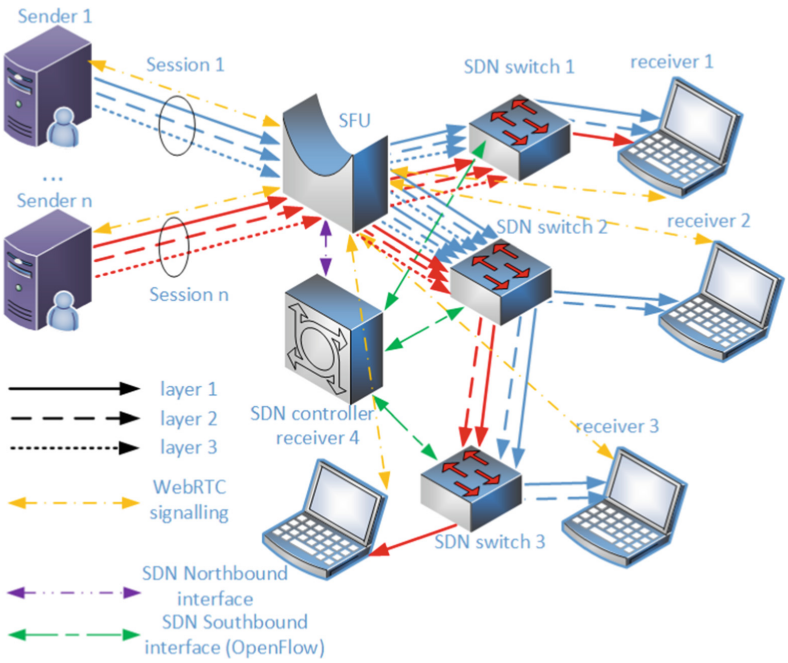

Fig. 2. General concept of the SDN-assisted layered simulcast.

The concept of an SDN-assisted layered simulcast is presented in Fig. 2. Senders performing the layered coding stream all the layers to the SFU, which multiplies them and forwards them non-selectively to the destinations. SDN switches in the path between the SFU and a given receiver block unnecessary

layers. To avoid flooding of the network with unnecessary layers, switches that block them (SDN switch 1 and SDN switch 2) should be located as close the SFU as possible, preferably on the same network node as the SFU.

Under stable network conditions, unnecessary layers are not sent beyond the first switch in the path from the SFU to a given receiver. When increasing congestion is detected, the switches in the path block the transmission of the highest received layer (unless it is the base layer). The switch near the congested node also will block this layer forwarding and no packets of this layer will be injected into the congested area, even though some of them are still forwarded (see switch 3 and receiver 4 in Fig. 2).

The SDN network management is handled by the SDN controller, which has knowledge of the network topology and the current state of the network. The SDN controller is connected via the Northbound interface to the SFU from which it obtains information about the current state of the streamed media. On this basis, the SDN controller calculates the optimal number of layers that can be sent on the path between the SFU and a given receiver. The controller also gives feedback to the SFU about the current state of the network. The calculations of the number of layers that will be transmitted on the path from the sender to the receiver are performed according to Algorithm 1.

In the proposed solution, the SFU only coordinates all conferences in one site, providing signaling capabilities. Each SDN switch can act as the traditional SFU which select and forward desired layers.

4 Experiments

The proposed system was implemented in the Jitsi Videobridge [38] and SDN infrastructure of the multi-Gigabit European Academic NeTwork (GEANT). The Geant offers the Testbeds as a Service (TaaS) [39] service that uses eight nodes (Fig. 3), one each in Amsterdam, Bratislava, Hamburg, London, Madrid, Milano, Paris, and Prague. Each node includes a router (used to communicate with other nodes), an SDN switch supporting the OpenFlow protocol, and a local switch (used to communicate inside the node).

For the purposes of our experiments, the resources (bare metal servers, virtual servers and network equipment) of five nodes have been reserved. The conference bridge and a bulk traffic generator were placed in Paris. The WebRTC senders were placed in Amsterdam, London, and Hamburg. The WebRTC receiver was placed in Madrid, and the receiver of bulk data that were the background Transmission Control Protocol (TCP) traffic, as well. The SDN controller was placed in Paris, as was the conference bridge, which simplified the communication between these devices. SDN switches were placed on all eight network nodes.

As the software of a conference bridge, the Jitsi Videobridge that implements the proposed extensions was used. For comparison, the original Jitsi Videobridge, without our extensions, was used. During all experiments, default time constants of the Jitsi Videobridge were set.

Algorithm 1. Calculate available layers on the path

Require: conference poarticipant destination $ConfDst$, conference sender (source) $ConfSrc$

Initialize
$P \leftarrow GetPath(ConfSrc, ConfDst)$
$L \leftarrow GetLayers(ConfSrc)$
$c_{min} \leftarrow \infty$
while $p_i \in P$ **do** /* calulate avaiable capcity on on the path */
    $c_i \leftarrow EstimateAviableCapacity(p_i)$
    if $c_i > c_{min}$ **then**
        $c_{min} \leftarrow c_i$
    end if
end while
$layers \leftarrow 0$
$r_{total} \leftarrow 0$
while $l_i \in L$ **do**
    if $(r_{total} + Rate(l_i)) < c_{min}$ **then**
        $layers \leftarrow layers + 1$
        $r_{total} \leftarrow r_{total} + Rate(l_i)$
    else
        return layers
    end if
end while

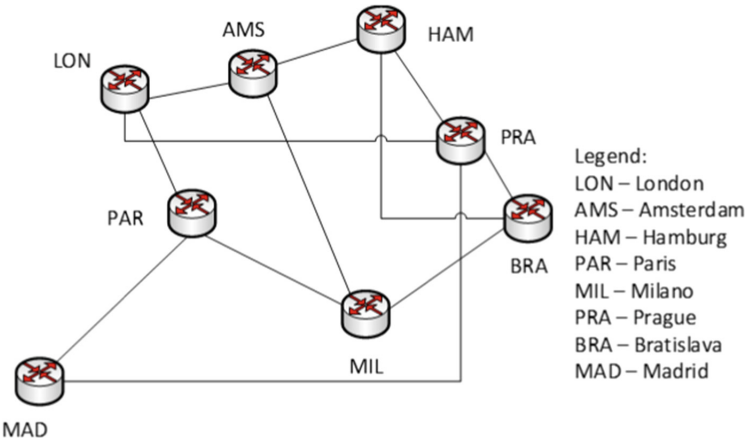

Fig. 3. Topology of GÉANT testbed. Distances between nodes have been preserved.

Video senders used the author's WebRTC application. Video from the camera was emulated by files of raw video information, both taken from a publicly available site [40] and captured as needed for this research (the latter includes the talking head of one of the authors, the parking lot of the AGH University, and a laboratory room). Video receivers used the author's WebRTC application.

For sake of comparison, the author's application of the receiver-driven layered multicast was also used. WebRTC applications used the Google Chrome browser as their run-time environment. In the case of the senders, Google Chrome was configured to use an external y4m file (video stream written in YUV4Mpeg format) as a video source instead of a camera. As the generator of TCP background traffic, the iperf tool [41] was used.

The links between the conference bridge and the senders were set to 100 Mbps, and the links between the bridge and the receivers were set to 2 Mbps or 10 Mbps (default for the GEANT testbed). The throughputs of the links between the bridge and the receivers were the same during a single experiment.

For the purposes of the evaluation of the proposed solution, three experiments were carried out. The first one was focused on static (infrastructural) limitations and their impact on a WebRTC layered simulcast. The next two were focused on dynamic constrains, introduced by layers competing for network resources (experiment 2) and by competing layers and TCP flows (experiment 3). The WebRTC senders had at their disposal 3 ($N = 3$) layers: the first one was the base layer, the second one was a spatial layer, and the third one was a temporal layer. The target bit rate (TBR) of the two first layers was equal to 0.6 Mbps, the third one to 1.2 Mbps, and the total TBR (i.e. TBR of the highest quality streaming video) was 2.4 Mbps.

5 Evaluation

The proposed solution was tested in three experiments, described in the previous section. Results of these experiments were compared with results obtained in corresponding experiments carried out in the same circumstances with the use of the WebRTC's receiver-driven layered simulcast.

5.1 Experiment 1: Single Video Streaming, No Background Traffic

Experiment 1 was aimed at checking the stability of the communication system and its ability to select the optimal number of simulcasted layers in the case of a lack of dynamic load. During the experiment, the layered sender placed in London streamed video to the SFU in Paris, and then the required layers were re-sent from the SFU in Paris to the receiver in Madrid. Distance between the bridge and the receiver was about 1200 km. This resulted in a propagation delay of about 6 ms. The throughput of links between the SFU and receivers was set to 2 Mbps, so the network was not able to transmit 2.4 Mbps of the total target bit rate. Because in this experiment only static (infrastructure) constraints should affect the layered simulcast, no background traffic was injected into the network, and simulcasted streams competed for throughput with themselves.

The results obtained using the proposed solution are shown in Fig. 4. About five seconds after starting the observation, Jitsi Videobridge switched on the base layer (layer 1), and about five seconds later, the spatial layer (layer 2) was also switched on (Fig. 4a). The total throughput of the first two layers is

approximately 1.200 Mbps. This is much less than the throughput of the link (2 Mbps), so no errors were detected (Fig. 4b).

Lossless transmission means that after exceeding the next time limit, the Jitsi Videobridge will decide to attach the next layer. At this point, a classic simulcast-based congestion control mechanism (which estimates the current state of the network through trial and error) would switch on the time layer (i.e. layer 3). This would inevitably lead to packet loss. However, our solution makes the decision about switching on/off any layer on the basis of both the network topology, the bit rate of the incoming layers, and the available throughput. As a result, our solution did not enable layer 3 (Fig. 4a), so the video transmission remained lossless (Fig. 4b). The zero packet loss shown in Fig. 4b proves the correct response of the proposed congestion control mechanism, which, having knowledge of the traffic and network, always switches on only those layers that allow for the transmission of the highest possible total bit rate, provided that the total transmission rate does not exceed the throughput of the path between the transmitter and the receiver.

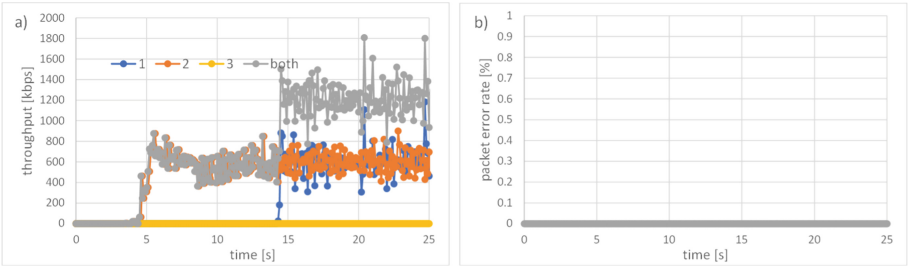

Fig. 4. Layered simulcast provided by the proposed solution: (a) throughput as a function of time, (b) packet error rate as a function of time.

For sake of comparison, the WebRTC simulcasting that use the original Jitsi Videobridge was tested in the same circumstances (Fig. 5). Due to the same Jitsi Videobridge settings, the first two layers were switched on at the same times (counting from the start of transmission) as in the case shown in Fig. 4. Since finding the available throughput is now a trial and error process, the temporal layer (layer 3) also was switched on. The cumulative throughput of the three layers exceeds 2 Mbps, which resulted in packet loss due to congestion. Although the layer 3 was soon switched off, unacceptably high (between 8 and 18 percent, median 12 percent) packet error rates (PERs) were observed for at least 5 s (Fig. 5b).

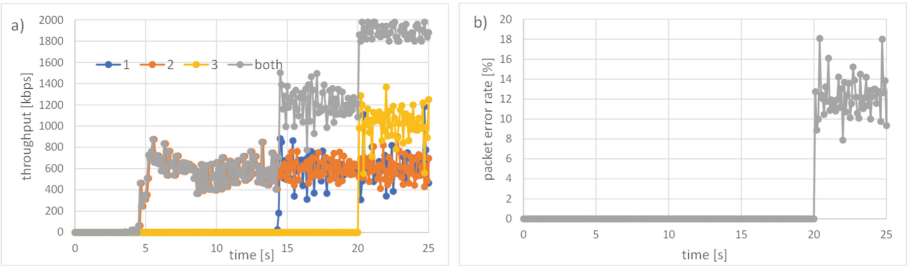

Fig. 5. Typical solution of WebRTC layered simulcast: (a) throughput as a function of time, (b) packet error rate as a function of time.

The trial-and-error evaluation of the available throughput means that, unlike the proposed solution, the layer 3 will be switched on every now and then, each time producing equally high error rates. Network neutrality, which results in non-discrimination of the transmitted layers, also resulted in packet loss not only in layer 3, but in all three layers. As a result, the quality of video transmission experienced by a user is unacceptable from time to time (in fact, the user then sees a still picture - the last properly received video frame).

5.2 Experiment 2: Multiple Video Streaming, No TCP Traffic

Experiments 2 was aimed at checking the system in terms of stability and the selection of layers in the presence of both static and dynamic constrains, while dynamic ones were introduced by multiple simulcast sessions. During this experiment, the receiver in Madrid participated in 1 to 5 simulcast sessions (each from 1 to 3 layers), established by the sender in London (1 session) and the senders in Amsterdam and Hamburg (each from 0 to 2 sessions). The throughput of links between the SFU and the receivers was set to 10 Mbps. The packet error rates that occurred during Experiment 2 are listed in Table 1.

When the number of WebRTC sessions was less than or equal to 2 (cumulative TBR less than half of 10 Mbps), no error was observed during transmission carried out with the use of both the proposed solution and the original Jitsi Videobridge (Table 1). Although the cumulative TBR of a single simulcast session (2.4 Mbps) should ensure the receipt of all 3 layers up to 4 sessions, the heterogeneity of footage available at [40] (movies with scene changes) causes temporary significant increase in traffic volume when changing scenes and then quick return to the set TBR. As a result, the increase in the number of sessions to 3 caused that the temporal layer (layer 3) was not able to be fully transmitted, and further increase in the number of sessions (to 4 and 5) caused both the spatial and temporal layers (layers 2 and 3) not to be fully transmitted through the network.

Table 1. Packet error rates observed in systems that used the original Jitsi Videobridge and the proposed solution. No TCP background traffic.

No. of WebRTC sessions	original			proposed		
	No. of simulcast stream within WebRTC session					
	1	2	3	1	2	3
1	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
2	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
3	0.0%	0.0%	5.1%	0.0%	0.0%	0.2%
4	0.0%	3.3%	11.4%	0.0%	0.1%	1.5%
5	0.0%	6.7%	20.2%	0.0%	0.4%	1.8%

This effect occurred when both the proposed solution and the original Jitsi Videobridge was used. However, when the proposed solution was used, the PER observed was one order of magnitude smaller than that of the original solution. In absolute terms, this resulted in an improvement of between 3.2 (4 sessions, 2 layers) and 18.4 (5 sessions, 3 layers) percentage points, and in relative terms, an improvement of 3 (4 sessions, 2 layers) to 13.2 (4 sessions, 3 layers) percent.

5.3 Experiment 3: Multiple Video Streaming, TCP Traffic

Experiment 3 extends the experiment 2 with presence of background bulk data traffic, transmitted with the use of the TCP. As a result, a given simulcast session has to adapt to static (infrastructural) limitations and to dynamic limitations introduced by both competing WebRTC sessions and TCP flows. The packet error rates that occurred during Experiment 3 are shown in Table 2.

Table 2. Packet error rates observed in systems that used the original Jitsi Videobridge and the proposed solution. TCP background traffic.

No. of WebRTC sessions	original			proposed		
	No. of simulcast stream within WebRTC session					
	1	2	3	1	2	3
1	0.0%	0.2%	0.5%	0.0%	0.1%	0.2%
2	0.1%	1.3%	3.2%	0.1%	0.2%	0.4%
3	0.1%	3.9%	7.3%	0.1%	0.3%	0.5%
4	0.1%	7.4%	11.5%	0.1%	0.4%	0.7%
5	0.2%	10.4%	15.2%	0.2%	0.4%	0.8%

Each session competed for link throughput with other sessions, as in experiment 2, and, additionally, with TCP traffic. As a result, error-free transmission was possible only in the case of a single base layer sharing the link with one TCP flow. In all other cases, the introduction of TCP background traffic caused that layers, including the base one, were not transmitted without errors.

When only the base layers were transmitted, simulcast congestion control was not possible, so the PERs only depended on the number of WebRTC sessions (the greater the number of sessions, the higher the PER), and was not dependent on simulcast management. As a result, the PERs observed when the proposed solution was used and the PERs observed when the original solution was used were equal to each other.

In the remaining cases, the packet error rate depended both on the number of WebRTC sessions (i.e. the number of layers competing for the throughput of the shared link) and the effectiveness of the congestion control mechanisms used. This time, the increase in PER as a function of the number of simulcast sessions, observed when the proposed solution was used, was significantly slower than the corresponding increase in PER observed when the original solution was used (Table 2). In particular, after using our solution, the packet error rate decreased two times for a single simulcast session, and for five simulcast sessions it decreased by two orders of magnitude. When the first two layers were switched on, the improvement was, in absolute terms, from 0.1 (1 session) to 10 (5 sessions) percentage points, and in relative terms, from 3.8 (5 sessions) to 50 (1 session) percent. When all three layers were switched on, the improvement was, in absolute terms, from 0.3 (1 session) to 14.4 (5 sessions) percentage points, and in relative terms, from 5.5 (5 sessions) to 40 (1 session) percent.

6 Conclusions

In this paper the SDN-assisted conference bridge acting as a SFU for a WebRTC layered simulcasting is proposed. Part of the SFU functionality (related to multiplication and forwarding of layered video) is accomplished by the bridge, while the other part (related to selection layers for forwarding purposes) is transferred to the SDN. During simulcast transmissions, the conference bridge gets data on the network state from the SDN controller, and the controller gets data on the state of the transmitted video from the bridge.

The results of the evaluation carried out in the GEANT testbed network showed that in the network not loaded with TCP traffic, but only with video traffic, packet error rates were up to 1.8%, and in the network loaded with TCP traffic, packet error rates were up to 0.8%. The worst-case error rates were observed for 5 simulcast sessions, each with 3 layers turned on, whose total target bit rate has exceeded the bottleneck throughput by 20%. In the same circumstances, the classic solution achieved PERs of 20.2%, and 15%, respectively.

In conclusion, the SDN controller included in the process of selecting layers depending on the current network state, is able to react quickly and autonomously. The use of both information about the network state and information about the WebRTC simulcast session sent from the SFU allowed the controller to avoid the misbehaviour of simulcasted transmissions when a new layer is switched on, caused by the overshooting of the estimation of available bandwidth, typical for classic solutions based on trial-and-error evaluation.

References

1. Loreto, S., Romano, S.P.: Real-Time Communication with WebRTC: Peer-to-Peer in the Browser. O'Reilly Media Inc. (2014)
2. Jennings, C., Boström, H., Bruaroey, J.: WebRTC 1.0: Real-Time Communication between Browsers; W3C Recommendation (2021). https://www.w3.org/TR/2021/REC-webrtc-20210126/. Accessed 15 Feb 2023
3. André, E., Le Breton, N., Lemesle, A., Roux, L., Gouaillard, A.: Comparative study of WebRTC open source SFUs for video conferencing. In: Proceedings of the 2018 Principles, Systems and Applications of IP Telecommunications (IPTComm), pp. 1–8 (2018)
4. López, L., et al.: Kurento: the WebRTC modular media server. In: Proceedings of the 24th ACM International Conference on Multimedia, pp. 1187–1191 (2016)
5. Chodorek, R.R., Chodorek, A., Rzym, G., Wajda, K.: A comparison of QoS parameters of WebRTC videoconference with conference bridge placed in private and public cloud. In: Proceedings of the IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), pp. 86–91 (2017)
6. Chodorek, A., Chodorek, R.R., Wajda, K.: An analysis of sender-driven WebRTC congestion control coexisting with QoS assurance applied in IEEE 802.11 wireless LAN. In: Proceedings of the 2019 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), pp. 1–5 (2019)
7. Singh, V., Lozano, A.A., Ott, J.: Performance analysis of receive-side real-time congestion control for WebRTC. In: Proceedings of the International Packet Video Workshop, pp. 1–8 (2013)
8. Chodorek, A., Chodorek, R.R., Wajda, K.: Comparison study of the adaptability of layered and stream replication variants of the WebRTC simulcast. In: Proceedings of the International Conference on Software, Telecommunications and Computer Networks (SoftCOM), pp. 1–6 (2019)
9. Grozev, B., Politis, G., Ivov, E., Noel, T., Singh, V.: Experimental evaluation of simulcast for WebRTC. IEEE Commun. Standards Mag. **1**(2), 52–59 (2017)
10. Loreto, S., Romano, S.P.: How far are we from WebRTC-1.0? An update on standards and a look at what's next. IEEE Commun. Mag. **55**(7), 200–207 (2017)
11. Uberti, J.: Simulcast encoding is now supported in @webrtc for the VP8, VP9, and H.264 codecs - try it out using the latest Chrome Canary, on Twitter (2019). https://twitter.com/juberti/status/1085764367113572353. Accessed 15 Feb 2023
12. Scalable Video Coding (SVC) Extension for WebRTC, W3C Editor's Draft (2023). https://w3c.github.io/webrtc-svc/. Accessed 15 Feb 2023
13. Kirmizioglu, R.A., Kaya, B.C., Tekalp, A.M.: Multi-party WebRTC videoconferencing using scalable VP9 video: from best-effort over-the-top to managed value-added services. In: Proceedings of the 2018 IEEE International Conference on Multimedia and Expo (ICME), pp. 1–6 (2018)
14. Kirmizioglu, R.A., Tekalp, A.M.: Multi-party WebRTC services using delay and bandwidth aware SDN-assisted IP multicasting of scalable video over 5G networks. IEEE Trans. Multimed. **22**(1), 1005–1015 (2019)
15. Kirmizioglu, R.A., Tekalp, A.M., Görkemli, B.: Distributed Virtual Selective-Forwarding Units and SDN-Assisted Edge Computing for Optimization of Multi-party WebRTC Videoconferencing (2022). Available at SSRN. https://ssrn.com/abstract=4045902 (Preprint submitted to Signal Processing: Image Communication). Accessed 15 Feb 2023

16. Holmberg, Ch., Eriksson, G., Hakansson, S.: Web Real-Time Communication Use Cases and Requirements. RFC 7478, IETF (2015)
17. Roach, A.: WebRTC Video Processing and Codec Requirements. RFC 7742, IETF (2016)
18. Valin, J.M., Bran, C.: WebRTC Audio Codec and Processing Requirements. RFC 7874, IETF (2016)
19. Romano, S.P., Roseti, C., Tulino, A.M.: SHINE: secure hybrid in network caching environment. In: Proceedings of the 2018 International Symposium on Networks, Computers and Communications (ISNCC), pp. 1–6 (2018)
20. Bakar, G., Kirmizioglu, R.A., Tekalp, A.M.: Motion-based adaptive streaming in WebRTC using spatio-temporal scalable VP9 video coding. In: 2017 IEEE Global Communications Conference, GLOBECOM 2017, pp. 1–6 (2017)
21. Petrangeli, S., Pauwels, D., van der Hooft, J., Wauters, T., De Turck, F., Slowack, J.: Improving quality and scalability of WebRTC video collaboration applications. In: Proceedings of the 9th ACM Multimedia Systems Conference, pp. 533–536 (2018)
22. Chodorek, A., Chodorek, R.R., Wajda, K.: Media and non-media WebRTC communication between a terrestrial station and a drone: the case of a flying IoT system to monitor parking. In: Proceedings of the 2019 IEEE/ACM 23rd International Symposium on Distributed Simulation and Real Time Applications (DS-RT), pp. 1–4 (2019)
23. Chodorek, A., Chodorek, R.R., Yastrebov, A.: The prototype monitoring system for pollution sensing and online visualization with the use of a UAV and a WebRTC-based platform. Sensors **22**(4), 1578 (2022)
24. Chodorek, A., Chodorek, R.R., Yastrebov, A.: Weather sensing in an urban environment with the use of a UAV and WebRTC-based platform: a pilot study. Sensors **21**(21), 7113 (2021)
25. Loreto, S., Romano, S.P.: Real-time communications in the web: Issues, achievements, and ongoing standardization efforts. IEEE Internet Comput. **16**(5), 68–73 (2012)
26. Blum, N., Lachapelle, S., Alvestrand, H.: WebRTC-realtime communication for the open web platform: what was once a way to bring audio and video to the web has expanded into more use cases we could ever imagine. Queue **19**(1), 77–93 (2021)
27. Amirante, A., Castaldi, T., Miniero, L., Romano, S.P.: On the seamless interaction between webRTC browsers and SIP-based conferencing systems. IEEE Commun. Mag. **51**(4), 42–47 (2013)
28. Johnston, A., Yoakum, J., Singh, K.: Taking on webRTC in an enterprise. IEEE Commun. Mag. **51**(4), 48–54 (2013)
29. Burman, F., Westerlund, M., Nandakumar, S., Zanaty, M.: Using Simulcast in Session Description Protocol (SDP) and RTP Sessions. RFC 8853, IETF (2021)
30. Lin, X., et al.: GSO-simulcast: global stream orchestration in simulcast video conferencing systems. In: Proceedings of the ACM SIGCOMM 2022 Conference, pp. 826–839 (2022)
31. Xhagjika, V., Escoda, O.D., Navarro, L., Vlassov, V.: Media streams allocation and load patterns for a WebRTC cloud architecture. In: Proceedings of the 8th International Conference on the Network of the Future (NOF), pp. 14–21 (2017)
32. Bakar, G., Kirmizioglu, R.A., Tekalp, A.M.: Motion-based rate adaptation in WebRTC videoconferencing using scalable video coding. IEEE Trans. Multimed. **21**(2), 429–441 (2018)

33. Romano, S.P., Giangrande, F.: On the use of network coding as a virtual network function in satellite-terrestrial CDNs. In: Proceedings of the IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 662–667 (2018)
34. Jero, S., Gurbani, V.K., Miller, R., Cilli, B., Payette, C., Sharma, S.: Dynamic control of real-time communication (RTC) using SDN: a case study of a 5G end-to-end service. In: Proceedings of the NOMS 2016–2016 IEEE/IFIP Network Operations and Management Symposium, pp. 895–900 (2016)
35. Cox, J.H., Clark, R., Owen, H.: Leveraging SDN and WebRTC for rogue access point security. IEEE Trans. Netw. Serv. Manag. **14**(3), 756–770 (2017)
36. Boubendir, A., Bertin, E., Simoni, N.: Network as-a-service: the WebRTC case: how SDN & NFV set a solid Telco-OTT groundwork. In: Proceedings of the 2015 6th International Conference on the Network of the Future (NOF), pp. 1–3 (2015)
37. Boubendir, A., Bertin, E., Simoni, N.: On-demand dynamic network service deployment over NaaS architecture. In: Proceedings of the NOMS 2016–2016 IEEE/IFIP Network Operations and Management Symposium, pp. 1023–1024 (2016)
38. Jitsi Videobridge. https://jitsi.org/jitsi-videobridge/. Accessed 15 Feb 2023
39. Farina, F., Szegedi, P., Sobieski, J.: GÉANT world testbed facility: federated and distributed testbeds as a service facility of GÉANT. In: Proceedings of the 26th International Teletraffic Congress (ITC), pp. 1–6 (2014)
40. Xiph.org Video Test Media [Derf's collection]. https://media.xiph.org/video/derf/. Accessed 15 Feb 2023
41. iPerf. https://iperf.fr/. Accessed 15 Feb 2023