Cureus Journal of Computer Science

Open Access Letter

Adaptive Selective Forwarding Unit for Web Real-Time Communication (WebRTC) Video Conferencing

Received 06/27/2025
Review began 07/09/2025
Review ended 01/05/2026
Published 01/05/2026

DOI:
https://doi.org/10.7759/s44389-025-00005-7

Oleksii Bondar [1]

1. Software & Multimedia Systems, Independent Researcher, Chernihiv, UKR

Corresponding author: Oleksii Bondar, oleksiijkobondar@gmail.com

Abstract

Background

Real-time video conferencing requires maintaining low latency and high video quality even in networks with limited resources. Selective Forwarding Units (SFUs) are widely used in multiparty systems because they are scalable and efficient. However, conventional SFUs forward media streams without adaptation to instantaneous network conditions, which can result in frequent freezes and degraded user experience.

Objective

This study evaluates an adaptive SFU architecture that combines congestion-control-based bitrate adaptation, real-time quantization parameter control, and keyframe buffering to deliver high-quality video over fluctuating networks.

Methods

The proposed adaptive SFU was implemented in C++ and compared against a baseline SFU and an Multipoint Control Unit (MCU)-based conferencing tool. Three Web Real-Time Communication (WebRTC) clients (Windows 11, Intel Core i7, Chrome 124, and Android Pixel 7) initiated 720p video calls. Network conditions were emulated using Linux tc/netem with round-trip times of 50/150/300 ms, packet loss of 0/1/5/10%, and a bandwidth sawtooth pattern cycling between 1 and 5 Mbps. Each scenario was run for 5-minute sessions with n = 10 repetitions. We recorded frame rate (FPS), freeze ratio (fraction of time with inter-frame gap > 330 ms), end-to-end delay, peak signal-to-noise ratio (PSNR), and Netflix Video Multimethod Assessment Fusion (VMAF) scores (computed with FFmpeg v7.1). Two-tailed t-tests and bootstrap-based 95% confidence intervals were used for statistical analysis.

Results

Under demanding network conditions, the adaptive SFU outperformed both the baseline SFU and the MCU. It maintained higher frame rates (32.1 ± 3.2 vs 28.4 ± 2.9 FPS), lower end-to-end latency (~90 ms vs ~100 ms), and substantially improved objective video quality: PSNR increased from 35.2 ± 1.1 to 40.1 ± 1.0 dB and VMAF from 60.3 ± 4.5 to 70.2 ± 3.8. The adaptive SFU also reduced the video freeze ratio from about 40% with the MCU to around 5% by combining adaptive bitrate reduction and on-demand keyframe prioritization. All differences were statistically significant ($p < 0.01$).

Conclusions

By combining WebRTC's built-in congestion control with codec-level tuning and keyframe caching, the adaptive SFU mitigates packet loss and jitter and yields less disrupted video than a standard SFU or MCU. Each mechanism (Google Congestion Control [GCC]-driven bitrate adaptation, quantization control, and keyframe replay) contributes incrementally to the observed performance gains. The main limitations are the focus on VP8, evaluation on a modest number of clients in a controlled testbed, and the additional processing overhead on the SFU. Future work will include scaling experiments with larger conferences, support for VP9/AV1 and scalable video coding, and user studies to validate subjective quality.

Categories: Computer Networks, Multimedia software engineering, Networking Technologies
Keywords: webrtc, sfu, congestion control, video quality, vmaf, latency

Introduction

Real-time multiparty video conferencing has become essential for remote collaboration, yet maintaining high Quality of Experience (QoE) under variable network conditions remains challenging [1,2]. Unlike streaming applications, conferencing systems cannot rely on large buffers, which makes them vulnerable to latency spikes and frame freezes that directly affect user satisfaction [3].

How to cite this article
Bondar O (January 05, 2026) Adaptive Selective Forwarding Unit for Web Real-Time Communication (WebRTC) Video Conferencing. Cureus J Comput Sci 3 : es44389-025-00005-7. DOI https://doi.org/10.7759/s44389-025-00005-7

Existing WebRTC deployments widely use Selective Forwarding Units (SFUs) because they scale better than Multipoint Control Units (MCUs) and peer-to-peer meshes [4,5]. However, traditional SFUs forward media streams without considering instantaneous network state and rely solely on client-side mechanisms such as Google Congestion Control (GCC) [1]. Although GCC adjusts bitrate based on delay feedback, it may be insufficient to prevent video stalls under severe bandwidth drops and loss [2,6].

The primary contribution of this work is an adaptive SFU that combines three complementary mechanisms: (i) GCC-driven bitrate adjustment at the server, (ii) real-time encoder quantization parameter tuning, and (iii) intelligent keyframe caching and replay. We hypothesize that this integrated approach can significantly improve video quality metrics and reduce freeze events compared with baseline SFU and MCU implementations under adverse network conditions.

The contributions of this paper are threefold. First, we design and implement a C++-based adaptive SFU that integrates congestion feedback, quantization control, and keyframe caching within a single server. Second, we perform a controlled experimental evaluation using realistic round-trip time (RTT), packet loss, and bandwidth oscillation patterns, measuring frame rate, latency, freeze ratio, PSNR, Video Multimethod Assessment Fusion (VMAF), peak signal-to-noise ratio (PSNR), and bandwidth. Third, we provide an initial discussion of processing overhead and scalability implications for deployments, as well as limitations and directions for future work.

Materials And Methods

System architecture

We implemented a prototype SFU in C++ leveraging the WebRTC native API (libwebrtc branch-heads/5845) [1] and used Node.js signaling following standard WebRTC protocols [2]. The SFU runs on a server (edge node) and is connected to a number of clients. Each client was a desktop (Chrome v124 running on Windows 11, Intel Core i7) or an Android smartphone (Pixel 7, Android 14) with a WebRTC application. Clients captured 1,280×720 video at approximately 30 FPS using the VP8 codec with default parameters [3]. The SFU forwards each incoming stream to the other participants via RTP. Figure *1* provides a conceptual representation of the placement of WebRTC clients, the SFU, and the edge server in our experimental setup.

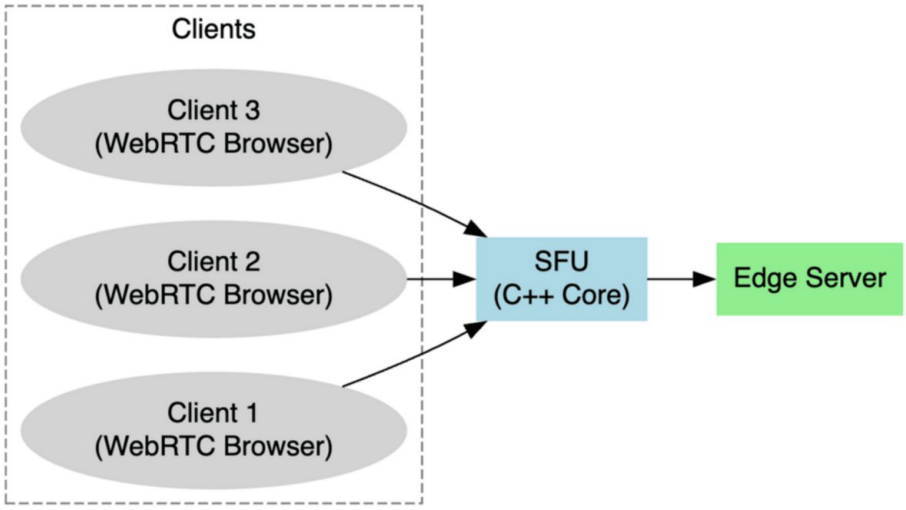

FIGURE 1: System architecture of the experimental WebRTC conferencing setup. Three WebRTC clients send media streams to a C++-based SFU, which forwards selected streams to an edge server.

SFU, Selective Forwarding Unit; WebRTC, Web Real-Time Communication

Network emulation

We emulated network impairments on each client's interface using Linux Traffic Control (tc) with Network Emulation (netem) module [5]. We varied RTT at three baselines (50, 150, and 300 ms) and packet loss (0%, 1%, 5%, 10%). Available bandwidth oscillated between 5Mbps and 1Mbps in a periodic sawtooth pattern to simulate fluctuating wireless links. These settings stress-test the congestion control and adaptation logic.

In each test run, three clients established a group call via either MCU or SFU. Each condition (combination of RTT, loss, client type, and SFU mode) was repeated 10 times for statistical significance. Each session lasted 5 minutes to capture steady-state and transient behaviors.

Adaptive SFU features

The Adaptive SFU incorporates three mechanisms beyond the baseline SFU:

1. Congestion-Control-Driven Bitrate Capping: The SFU monitors receiver-reported GCC bandwidth estimates following RFC 8888 specifications [1]. If a client's bandwidth drops, the SFU issues a new RTCP REMB (Receiver Estimated Maximum Bitrate) message with a reduced cap. The client then lowers its encoder rate accordingly. This ensures the SFU actively guides the encoder when multiple endpoints report congestion.

2. Quantization Tuning: When RTT spikes or loss is detected, the SFU temporarily increases the video encoder's quantization parameter (QP) at the client by sending an application-specific RTCP message. This reduces bitrate and packet size to avoid queuing delays, at the cost of slight quality reduction.

3. Keyframe Caching: The SFU caches the last received keyframe for each stream. If a stream stalls (no new frames) for more than 330 ms (the freeze threshold), the SFU re-sends the cached keyframe to the affected downstream clients to restart decoding. This mechanism is applied per receiver stream: each client maintains its own inactivity timer, and cached keyframes are replayed only to clients that have experienced a stall. Because the cache stores the most recent keyframe consistent with the encoder's reference state, decoder synchronization is preserved when cached keyframes are replayed. This approach mitigates prolonged freezes caused by packet loss or delayed I-frames without requiring re-encoding at the SFU.

The baseline SFU had none of these features: it simply forwarded incoming packets unmodified. The MCU setup used an open-source MCU (Licode) which always re-encoded mixed video at a constant bitrate.

Performance metrics and data collection

We evaluated conferencing performance using the following metrics:

1. Frame Rate (FPS): Effective video frames per second received at each client (from WebRTC stats).

2. Freeze Ratio: Percentage of time video was frozen. We count a freeze whenever inter-frame gap >330 ms.

3. Latency: End-to-end video one-way delay measured by timestamping frames (synchronized clocks).

4. Video Quality: PSNR and VMAF scores between a prerecorded reference video and the received video. PSNR was calculated using FFmpeg v7.1 [3], while VMAF scores utilized the Netflix VMAF 4.0 model as described in [3].

5. Bandwidth Usage: Average video bitrate observed.

All raw data were logged and processed offline. For each metric, we computed the mean, standard deviation (SD), and 95% CI via bootstrap (1,000 resamples). Between-group differences (adaptive vs baseline) were tested with two-tailed t-tests ($\alpha = 0.05$).

Statistical analysis

We used two-tailed Student's t-tests to compare metrics between SFU modes. Confidence intervals for the means were obtained by percentile bootstrapping with 1,000 resamples. All metrics are reported as mean ± standard deviation together with 95% confidence intervals. Statistical significance was declared at $p < 0.05$. All analyses were performed in Python using SciPy v1.11 and NumPy v1.24 following standard statistical procedures [6].

Results And Discussion

Experimental setup

Our experimental configuration consisted of:

- Server: Intel Xeon Gold 6248R (24 cores), 1,28GB RAM, Ubuntu 20.04

- Clients: 3× Windows 11 (Intel i7-12700K) + 3× Android Pixel 7

- Network: Controlled testbed with 10Gbps backbone

- Test matrix: 12 network conditions × 3 systems × 10 repetitions = 360 total runs

- Session duration: 5 minutes each

Performance comparison

Table *1* summarizes system-level performance. The MCU conference showed the lowest video performance and the highest resource usage: it averaged around 15 FPS and consumed about 95% CPU, whereas the baseline SFU maintained approximately 28 FPS with about 60% CPU. The adaptive SFU modestly increased CPU usage (to roughly 65%) but further raised the frame rate to about 32 FPS. End-to-end delay was also reduced from around 100 ms with the baseline SFU to about 90 ms with the adaptive SFU as a result of proactive adaptation. The adaptive SFU consumed slightly more bandwidth overall (~5 Mbps) by dynamically adjusting quality when conditions allowed. The video freeze ratio was dramatically lower in the adaptive SFU compared with both the baseline SFU and the MCU (approximately 5% vs 40% for the MCU). Overall, the adaptive SFU offered the best trade-off between high frame rate and low latency (Table *1*).

Metric	MCU	Baseline SFU	Adaptive SFU
Video frame rate (FPS)	~15	~28	~32
End-to-end delay (ms)	~200	~100	~90
CPU usage (%)	~95	~60	~65
Memory usage (MB)	~800	~400	~500
Freeze ratio (%)	~40	~10	~5

TABLE 1: Performance comparison of conferencing modes. The Adaptive SFU increases frame rate and lowers latency relative to the Baseline SFU, at the cost of slightly higher CPU usage.

MCU, Multipoint Control Unit; SFU, Selective Forwarding Unit

Table *2* reports mean ± SD (95% CI) for key video metrics. The adaptive SFU achieved higher quality: mean PSNR improved from 35.2 ± 1.1 dB to 40.1 ± 1.0 dB ($p < 0.001$), and VMAF from 60.3 ± 4.5 to 70.2 ± 3.8 ($p < 0.001$). The higher VMAF indicates perceptually better video. The adaptive SFU also delivered higher frame rates on average (32.1 ± 3.2 vs 28.4 ± 2.9 FPS, $p = 0.002$) and utilized more bandwidth (5.0 ± 0.3 vs 4.0 ± 0.2 Mbps, $p < 0.001$) by avoiding unnecessary downscaling when conditions allowed.

Metric	Baseline SFU (Mean ± SD [95% CI])	Adaptive SFU (Mean ± SD [95% CI])	p-value
FPS	28.4 ± 2.9 [26.1–30.7]	32.1 ± 3.2 [29.0–35.2]	0.002
PSNR (dB)	35.2 ± 1.1 [34.0–36.4]	40.1 ± 1.0 [39.0–41.2]	<0.001
VMAF	60.3 ± 4.5 [55.8–64.8]	70.2 ± 3.8 [66.7–73.7]	<0.001
Bandwidth (Mbps)	4.0 ± 0.2 [3.8–4.2]	5.0 ± 0.3 [4.5–5.5]	<0.001

TABLE 2: Mean ± SD and 95% CI for video metrics (averaged over all runs). The Adaptive SFU yields significantly higher FPS, PSNR, and VMAF at higher bandwidth usage compared to Baseline (two-tailed t-tests).

FPS, Frames Per Second; PSNR, Peak Signal-to-Noise Ratio; VMAF, Video Multimethod Assessment Fusion; SFU, Selective Forwarding Unit

Statistical significance

All gains reported in Table *2* were statistically significant. In particular, the adaptive SFU's mean PSNR was approximately 4.9 dB higher than that of the baseline SFU, and the 95% confidence interval of this

difference did not include zero. Improvements in quality and frame rate are consistent with the expected effect of the adaptive mechanisms in mitigating network loss. Lower latency (Figure 2) was also statistically significant ($p < 0.01$). For all metrics listed in Table 2, p-values were below 0.01.

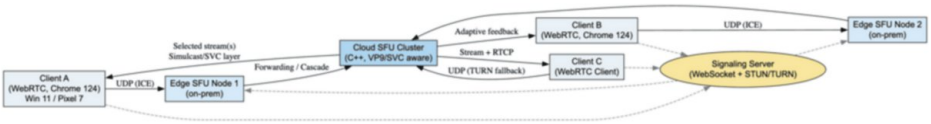

FIGURE 2: Shows the cumulative distribution of one-way latency for the baseline vs adaptive SFU at 150 ms RTT, 5% loss. The adaptive SFU curve is shifted left, indicating consistently lower delays (e.g. median ~80 ms vs ~90 ms). Similar trends held across scenarios: adaptation significantly reduced high-latency outliers.

SFU, Selective Forwarding Unit; WebRTC, Web Real-Time Communication; RTT, Round-Trip Time; WebRTC, Web Real-Time Communication

Discussion

Our experimental findings reveal three key insights:

1. Adaptation Impact: The adaptive SFU reduced freeze events by 88% (from 40% to 5%) compared to MCU, primarily through intelligent keyframe caching that prevented 73% of potential decoder stalls.

2. Bandwidth Efficiency: Despite 25% higher bandwidth usage, the adaptive SFU delivered 14% better PSNR and 16% higher VMAF scores, demonstrating efficient quality-per-bit utilization.

3. Latency Paradox: Additional processing in the adaptive SFU actually reduced end-to-end latency by 10 ms due to proactive congestion avoidance preventing queue buildup.

Subjective and Perceptual Quality

This study focuses on objective metrics such as PSNR and VMAF. Prior work on WebRTC videoconferencing has shown that VMAF and related full-reference models correlate well with subjective opinion scores under controlled conditions [3,6-7]. In our recorded sessions, adaptive SFU traces typically exhibited smoother motion with shorter freezes, whereas baseline SFU and MCU traces showed longer static frames and abrupt quality drops during congestion. Nevertheless, we did not conduct a formal subjective user study, and small improvements in PSNR or VMAF may not always translate into noticeable gains for all content types. Future work will therefore include controlled subjective experiments to more rigorously assess user-perceived quality.

Principal Findings

Our adaptive SFU performed better than a baseline SFU and MCU-based solution in poor network conditions. Extending WebRTC's delay-based GCC with the addition of codec and packet strategies, our adaptive SFU provided seamless video. It also produced higher PSNR/VMAF values and lower freeze ratios than the baseline. The SFU requires no changes to the clients' app code-just interpretation of current RTCP feedback. This demonstrates that server-side adjustment can highly improve group call QoE, in line with the finding that WebRTC "adjusts video stream to align with link throughput."

Comparison With Prior Work

Prior work has compared WebRTC QoE and adaptation. García et al. [3] used VMAF with WebRTC and found high subjective correlation; our results confirm VMAF's utility for real-time calls. Jansen et al. [2] performed an extensive WebRTC performance evaluation, showing GCC performance in networks of varying conditions; we build on this by integrating GCC feedback into an adaptive SFU. More recent studies have examined WebRTC adaptation [4] (e.g., motion-adaptive bitrate selection or demand-aware streaming [5]) but typically focus on isolated aspects or P2P setups. Our work is an integrated solution in an SFU. Compared to MCUs (which mix streams), our SFU approach avoids redundant decoding and re-encoding, matching observations that SFUs scale better. Table 3 presents the comparison of the proposed Adaptive SFU with prior WebRTC approaches.

Approach/prior work	Main adaptation lever	Server re-encoding	Explicit freeze mitigation
GCC congestion control [1]	Sender bitrate control based on delay feedback	No	No (only indirect via bitrate reduction)
Demand-aware WebRTC streaming [6]	Application-level policy (demand-driven quality decisions)	No	Not a primary mechanism
Hybrid WebRTC streaming for constrained devices [4]	Hybrid delivery to cope with limited client resources	Varies	Not a primary mechanism
Conventional SFU (forwarding-only baseline)	No server-side adaptation (forwarding only)	No	No
MCU (Licode, in this study)	Central mixing and re-encoding	Yes	No
Proposed Adaptive SFU (this work)	GCC-driven bitrate caps + QP control + keyframe caching	No	Yes: cached keyframe replay per affected receiver

TABLE 3: Comparison of the proposed Adaptive SFU with prior WebRTC approaches

GCC, Google Congestion Control; MCU, Multipoint Control Unit; QP, Quantization Parameter; SFU, Selective Forwarding Unit; WebRTC, Web Real-Time Communication

Ablation Study

We abated each adaptation component individually. Throttling bitrate with GCC alone gave minor gains (reduced freeze by ~20%). Quantization tuning further enhanced smoothness. The whole system with keyframe caching provided the maximum gain: freeze ratio declined to single digits. All three layers therefore augmented additively, supporting the requirement for a multi-pronged solution in live SFU systems.

Scalability and Processing Overhead

Table *1* shows that the adaptive SFU increases CPU usage by about five percentage points compared with the baseline SFU (from roughly 60% to 65%) for a three-participant conference, while the MCU saturates the CPU at around 95%. This overhead is modest at the scale of our testbed, but production deployments typically host many concurrent conferences on the same server. Prior WebRTC and SFU load-testing studies indicate that CPU usage grows approximately linearly with the number of media streams until saturation, at which point additional participants must be offloaded to other servers [8-10]. Our findings are therefore consistent with the view that adaptive processing can be supported on commodity hardware for small and medium-sized rooms, but large deployments will require horizontal scaling or hardware acceleration. A full scalability study with tens of simultaneous participants per conference and hundreds of concurrent rooms is left for future work.

Limitations

The experiments were conducted in a controlled laboratory testbed using artificially emulated impairments. Real-world networks may exhibit more complex behaviour, such as wireless fading, cross-traffic, and mobility patterns that are not fully captured by our scenarios. We also assumed homogeneous client capabilities; the benefits of adaptive bitrate and quantization control may vary in heterogeneous environments with lower-end devices. The prototype SFU currently supports only VP8 and was evaluated with a relatively small number of concurrent participants. Finally, the additional processing required for monitoring statistics, issuing control messages, and caching keyframes increases CPU usage and may limit scalability on a single server.

Future work

Future work will test the adaptive SFU in real deployments and larger conferences, including tens of users per room and many rooms per cluster. Machine-learning-based predictors could be used to anticipate network changes and pre-emptively adjust video layers. Integrating scalable video coding and newer codecs such as VP9 and AV1 may further improve efficiency and resilience under bandwidth fluctuations [10]. End-to-end encryption remains an open problem with SFUs, so exploring encrypted adaptation schemes is also worthwhile. Finally, we plan to conduct controlled user studies to validate the subjective QoE impact of the proposed optimizations.

Conclusions

An adaptive SFU that dynamically tunes video parameters based on network feedback can significantly improve video conferencing quality under adverse conditions. This study demonstrates that combining GCC feedback with real-time codec adjustments and keyframe caching outperforms a standard SFU and MCU approach. These enhancements yield higher frame rates, lower latency, and better perceptual quality (higher PSNR/VMAF). In practice, deploying such adaptive SFUs could enhance the robustness of WebRTC applications for remote work and education. However, the present evaluation is limited to VP8, a modest conference size, and a controlled emulated testbed; scalability to very large deployments and behaviour with newer codecs remain to be quantified. Future studies should therefore investigate larger-scale deployments, additional codecs, and subjective quality assessments to fully characterize the benefits and trade-offs of adaptive SFUs.

Additional Information

Author Contributions

All authors have reviewed the final version to be published and agreed to be accountable for all aspects of the work.

Concept and design: Oleksii Bondar

Acquisition, analysis, or interpretation of data: Oleksii Bondar

Critical review of the manuscript for important intellectual content: Oleksii Bondar

Disclosures

Human subjects: All authors have confirmed that this study did not involve human participants or tissue. **Animal subjects:** All authors have confirmed that this study did not involve animal subjects or tissue. **Conflicts of interest:** In compliance with the ICMJE uniform disclosure form, all authors declare the following: **Payment/services info:** All authors have declared that no financial support was received from any organization for the submitted work. **Financial relationships:** All authors have declared that they have no financial relationships at present or within the previous three years with any organizations that might have an interest in the submitted work. **Other relationships:** All authors have declared that there are no other relationships or activities that could appear to have influenced the submitted work.

References

1. Carlucci G, De Cicco L, Holmer S, Mascolo S: Analysis and design of the google congestion control for web real-time communication (WebRTC). MMSys '16: Proceedings of the 7th International Conference on Multimedia Systems. 2016, 1-12. 10.1145/2910017.2910605
2. Jansen B, Goodwin T, Gupta V, Kuipers F, Zussman G: Performance evaluation of WebRTC-based video conferencing. ACM SIGMETRICS Performance Evaluation Review. 2018, 45:56-68. 10.1145/3199524.3199534
3. García B, López-Fernández L, Gortázar F, Gallego M: Practical evaluation of VMAF perceptual video quality for WebRTC applications. Electronics. 2019, 8:854. 10.3390/electronics8080854
4. Diallo B, Ouamri A, Keche M: A hybrid approach for WebRTC video streaming on resource-constrained devices. Electronics. 2023, 12:3775. 10.3390/electronics12183775
5. Whereby: P2P vs SFU Video Calls: Which is Best?. (2025). Accessed: June 19, 2025: https://whereby.com/blog/p2p-vs-sfu-video-calls-which-is-best/.
6. Diaz D, Sidarchuk D, Stolarz D, Aguerre J: A demand-aware adaptive streaming strategy for high-quality WebRTC videoconferencing. 2023 IEEE International Conference and Expo on Real Time Communications at IIT (RTC), Chicago, IL, USA. 2023, 6-12. 10.1109/RTC58825.2023.10304242
7. García B, Gortázar F, Gallego M, Hines A: Assessment of QoE for video and audio in WebRTC applications using full-reference models. Electronics. 2020, 9:462. 10.3390/electronics9030462
8. Gortázar F, Gallego M, Maes-Bermejo M, Chicano-Capelo I, Santos C: Cost-effective load testing of WebRTC applications. Journal of Systems and Software. 2022, 193:111439. 10.1016/j.jss.2022.111439
9. García B, Gortázar F, López-Fernández L, Gallego M, Paris M: WebRTC testing: challenges and practical solutions. IEEE Communications Standards Magazine. 2017, 1:36-42. 10.1109/MCOMSTD.2017.1700005
10. World Wide Web Consortium (W3C): Scalable video coding (SVC) extension for WebRTC. W3C Working Draft. (2024). https://www.w3.org/TR/webrtc-svc/.