e-ISSN : 2722-4368

Analysis of WebRTC Video Streaming Scalability in a Broadcaster-Viewer Model Using Scenario-Based Load Testing

Achmad Torikul Huda[1*] , Doni El Rezen Purba[1], Roy Nuary Singarimbun[1], Muhammad Chaizir[1], Kevin Ilham Apriandy[2]
[1]Politeknik Negeri Media Kreatif, Jl. Srengseng Sawah Raya No.17, RT.8/RW.3, Srengseng Sawah, Kec. Jagakarsa, Kota Jakarta Selatan, Daerah Khusus Ibukota Jakarta 12630
[2]Politeknik Internasional Tamansiswa Mojokerto, Jl. Taman Siswa No.30, Mergelo, Purwotengah, Kec. Magersari, Kota Mojokerto, Jawa Timur 61311
*torik@polimedia.ac.id

Abstrak. *This study assesses the Quality of Service (QoS) performance of a Web Real-Time Communication (WebRTC)-based video streaming system operating on a local network utilizing a broadcaster-viewer model. The system was constructed with Node.js as the local server, accompanied by broadcaster and viewer web pages that facilitate laptop camera and microphone streaming, video conferencing, camera effects, and MP4 media playback. The research strategy utilized experimental performance evaluation through a scenario-based load testing approach and Quality of Service benchmarking. Testing was performed across three scenarios: one broadcaster with one viewer, one broadcaster with two viewers, and one broadcaster with three viewers. The examined QoS characteristics encompassed throughput, packet loss, delay, and jitter, assessed utilizing Wireshark. The test findings indicated that throughput escalated with the increase in viewers, rising from 2.525 Mbps in Scenario 1 to 5.026 Mbps in Scenario 3. Packet loss remained minimal throughout all scenarios, however it increased to 0.2% in Scenario 3. Scenario 2 had the largest average delay at 751.92 ms, whilst Scenario 3 demonstrated the lowest average delay at 349.06 ms and the minimal average jitter at 1.645 ms. The findings demonstrate that WebRTC can provide multi-viewer local streaming with steady performance; yet, a rise in watchers necessitates meticulous management of bandwidth and network stability to preserve video content quality.*

Keyword: WebRTC, video streaming, Quality of Service, throughput, delay, jitter, packet loss, Wireshark.

INTRODUCTION

The demand for rapid, adaptable digital communication that does not necessitate supplementary program installations has rendered browser-based real-time communication an increasingly prevalent solution. A critical technology facilitating this requirement is Web Real-Time Communication (WebRTC). WebRTC is an open standard that delineates a collection of ECMAScript/WebIDL-based Application Programming Interfaces (APIs) for the real-time transmission of media and application data between browsers or devices [1]. This technology is accessible in contemporary browsers and cross-platform native clients, enabling developers to create audio, video, and data communication systems without need on supplementary plugins [2], [3].

The COVID-19 pandemic has accelerated digital transformation, leading to the widespread adoption of virtual communication platforms in sectors like as education, the workplace, healthcare, and public services. Cisco anticipates that video traffic would constitute a predominant segment of worldwide internet traffic, including 82% of consumer internet traffic by 2022 [4], [5]. This scenario highlights the necessity for a real-time communication system that is both readily available and customizable to fulfill an organization's particular requirements. WebRTC is pertinent in this context since it facilitates low-latency, secure, and adaptable communication for live streaming, web conferencing, online collaboration, and internal communication systems[6].

The potential for WebRTC development in web streaming is becoming increasingly significant, as this technology facilitates the creation of more interactive, adaptable, and seamlessly connected streaming systems with web-based services. In contrast to traditional streaming models, which are predominantly unidirectional, WebRTC facilitates the creation of bidirectional or multiparty streaming services, including video conferencing, interactive online courses, remote

consultations, internal organizational live broadcasts, and browser-based multimedia collaboration platforms[7]. Research on WebRTC is essential for establishing the technical framework necessary to comprehend the stable operation of streaming systems amidst increasing user numbers, diverse media formats, and the simultaneous implementation of extra features such as camera effects or MP4 video playback[8].

The influence of WebRTC research and development transcends technical networking dimensions, encompassing the wider progression of the digital services ecosystem. Research findings on WebRTC performance can provide a basis for developing online streaming systems that are more efficient in infrastructure, simpler to install, and more autonomous, as they do not consistently depend on third-party platforms[9]. Moreover, assessing Quality of Service (QoS) characteristics in WebRTC is essential for identifying system capacity constraints, bandwidth necessities, transmission reliability, and the quality of video content available to users. Consequently, WebRTC research presents significant potential for advancing the creation of contemporary streaming platforms that are scalable, secure, and compatible with current digital communication requirements[10].

WebRTC employs a synthesis of media and network protocols engineered for real-time communication. Audio and video media in WebRTC are conveyed using the Real-time Transport Protocol (RTP), secured using Secure RTP (SRTP) and DTLS-SRTP, whereas WebRTC data channels employ SCTP over DTLS and UDP[11]. This approach facilitates quick communication; however, the video quality experienced by consumers is significantly affected by network circumstances. The quality of service factors, including throughput, packet loss, delay, and jitter, directly influence the quality of video content, encompassing frame smoothness, visual clarity, audio-video synchronization, and the minimization of freezing or stuttering[12].

Prior studies have established that WebRTC possesses significant promise for real-time communication and can be utilized in several applications beyond traditional audio and video [13]. It demonstrates that bandwidth constraints impact WebRTC quality of service, specifically with latency and jitter. Nevertheless, investigating WebRTC within a local broadcaster-viewer framework, incorporating diverse viewer counts, camera effects, video conferencing, and MP4 streaming necessitates additional research, particularly to statistically evaluate their influence on QoS metrics. This study evaluates the performance of a Node.js-based WebRTC video streaming system under three viewer scenarios: one viewer, two viewers, and three viewers, utilizing Wireshark for measurement.

LITERATURE REVIEW

1. Architecture and Transmission Mechanisms

The WebRTC architecture is structured as a tiered protocol stack that facilitates real-time communication directly via a web browser, eliminating the need for supplementary software. WebRTC categorizes communication methods into two primary channels: the signaling channel and the media/data transmission channel. The signaling channel may employ technologies including WebSocket, Server-Sent Events (SSE), or XMLHttpRequest (XHR), which operate over HTTP/HTTPS and TCP[14]. This channel facilitates the sharing of preliminary connection information, including the Session Description Protocol (SDP), ICE candidates, codec capabilities, and network settings between peers. This study employed Node.js as a local server to facilitate the signaling mechanism between the broadcaster and viewer prior to the establishment of the WebRTC media session[15].

WebRTC employs UDP-based protocols for media transmission to reduce communication latency. Audio and video material are conveyed using RTCPeerConnection utilizing the Real-time Transport Protocol (RTP), safeguarded by the Secure Real-time Transport Protocol (SRTP) and Datagram Transport Layer Security (DTLS). WebRTC employs the Interactive Connectivity Establishment (ICE) technique, augmented by STUN and TURN, to facilitate peer-to-peer connections on networks with NAT or firewalls[16]. STUN is utilized to ascertain a device's public

address, whereas TURN functions as a relay when a direct peer-to-peer connection is unfeasible. The RTCDataChannel facilitates non-media data transmission with SCTP over DTLS and UDP, hence enabling both reliable and unreliable data communication as dictated by the application's requirements[17].

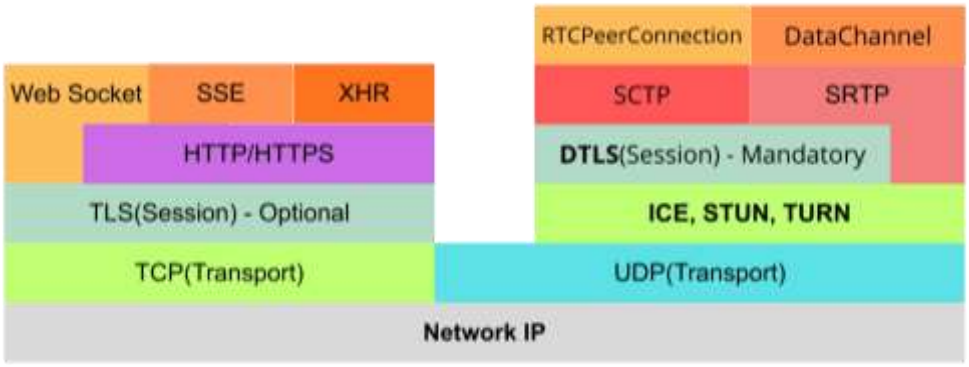

Figure 1. WebRTC Protocol Architecture

This architecture illustrates that WebRTC encompasses not only audio and video transmission but also offers a secure, adaptable, and low-latency data communication framework[18]. Nonetheless, despite WebRTC's intended for real-time communication, service quality is significantly influenced by network circumstances. Throughput, delay, jitter, and packet loss are critical parameters that influence the quality of video streaming, audio-video synchronization, and the stability of content received by users. Consequently, comprehending the WebRTC architecture is crucial prior to assessing system performance through Quality of Service (QoS) metrics[2].

2. Quality of Service Parameters in Multimedia Communications

Quality of Service (QoS) is a methodology for assessing network service quality through quantitative metrics. In multimedia communications, frequently utilized parameters encompass throughput, packet loss, latency, and jitter. ITU-T G.114 offers guidelines regarding the influence of one-way delay on communication quality[8], The TIPHON standard recognizes packet loss and delay jitter as critical determinants in the deterioration of terminal and network quality[9]. In the realm of WebRTC video streaming, QoS functions as both a metric of network performance and a gauge of the quality of the received video content, as subpar QoS values can lead to video interruptions, unstable frames, diminished image quality, and disrupted audio-video synchronization.

3. Throughput, Delay, Jitter, and Packet Loss

Throughput represents the actual data rate that was effectively transferred or received during the observation interval. Increased throughput enhances the system's capacity to provide video material, particularly when resolution and viewer numbers rise. Delay refers to the duration required for a packet to go from the source to the destination. Significant latency can result in unresponsive interactions during video conferences. Jitter refers to the fluctuation in latency among packets; elevated jitter leads to irregular packet arrival, potentially resulting in disrupted visual and audio[19]. Packet loss denotes the proportion of packets that are lost during transmission. In video streaming, packet loss may result in visual distortions, freezing, diminished frame quality, or audio segment loss. Consequently, the reliability of networks in WebRTC systems is predominantly influenced by the equilibrium among throughput capacity, low latency, regulated jitter, and minimal packet loss[7].

RESEARCH METHODOLOGY

This research employs an experimental performance assessment technique to evaluate the efficacy of a WebRTC-based video streaming system within a local network. The assessment was performed statistically by contrasting three viewer-count situations and thereafter evaluating the outcomes using QoS measures, such as throughput, packet loss, latency, and jitter. This methodology

was selected to evaluate the WebRTC system's capacity to sustain transmission quality when the number of stream recipients escalates.

1. Experimental System Design

The system being evaluated comprises a single broadcaster page and one or several viewer pages. The broadcaster functions as a media provider utilizing a camera, microphone, and MP4 video sources, whilst the viewer serves as a stream recipient. The system was constructed utilizing HTML, PHP, JavaScript, CSS, and Node.js as the local signaling server. The evaluated features encompass camera and microphone video streaming, camera effects, video conferencing, and MP4 video playback accessible by many viewer devices.[16].

2. Scenario-Based Load Testing

Load testing was performed via a scenario-based methodology, which entailed incrementally augmenting the viewer count to assess variations in system performance. Scenario 1 features a single broadcaster and one viewer; Scenario 2 includes a single broadcaster and two viewers; and Scenario 3 comprises a single broadcaster and three viewers. The influx of viewers is regarded as an escalation in connection load, since the broadcaster is required to disseminate media to a greater number of devices. This approach enables a quantifiable analysis of the system's initial scalability on QoS parameters.

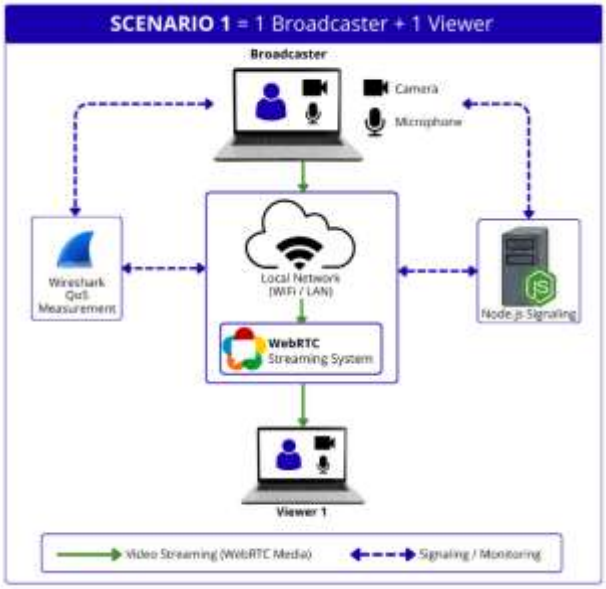

Figure 2. Scenario 1: one broadcaster and one viewer

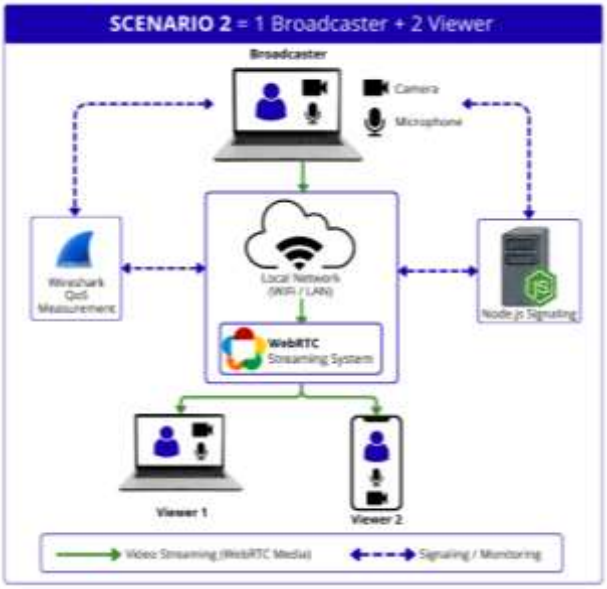

Figure 3. Scenario 2: one broadcaster and two viewers

Figure 2 illustrates tests conducted with a single broadcaster and a single viewer. This scenario served as a baseline to assess the initial performance of the WebRTC system with a single stream recipient. Under these circumstances, the transmission load is comparatively little as the broadcaster transmits media to a single device. The outcomes of Scenario 1 establish a standard for evaluating performance variations as the viewer count escalates in other situations.

Figure 3 illustrates a test with one broadcaster and two viewers. This scenario seeks to evaluate the impact of incorporating an additional receiver device on system performance. The system must manage media dissemination to many devices concurrently with two viewers. This scenario may elevate throughput demands and potentially influence delay, jitter, and packet loss metrics. Consequently, Scenario 2 is employed to assess the system's ability to sustain transmission quality while the connection load escalates.

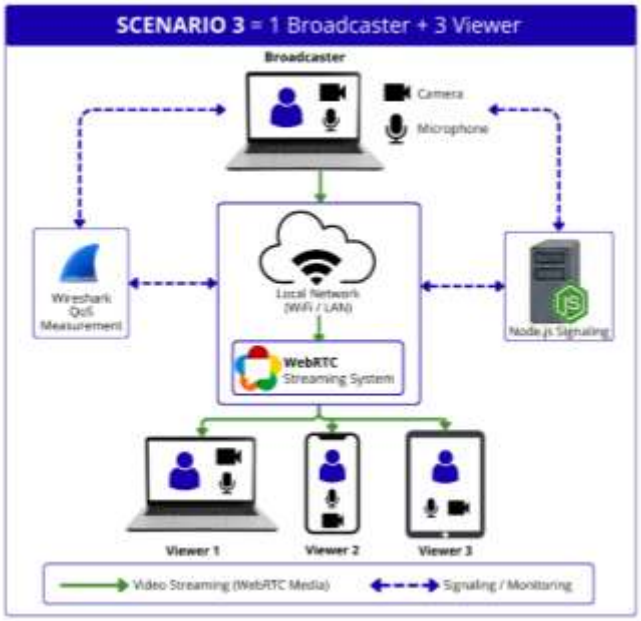

Figure 4. Scenario 3: one broadcaster and three viewers

In Figure 4 illustrates a test conducted with a single broadcaster and three viewers. This scenario exemplifies the maximum load circumstances in this study. The addition of three viewers was employed to assess the WebRTC system's capacity for multi-viewer media distribution within a local network. This scenario is crucial for assessing whether an increase in recipients leads to a deterioration in service quality, specifically regarding packet loss, delay, and jitter, which directly influence video fluidity, frame stability, and audio-video synchronization.

3. Packet-Level QoS Measurement

Measurements were performed utilizing a packet-level network measuring technique with the aid of Wireshark. Network traffic was filtered according to the IP addresses of the broadcaster and viewer, in addition to the UDP protocol utilized in the WebRTC session[20]. The acquired data was utilized to compute throughput, packet loss, delay, and jitter, considering the byte count, packet count, transmission duration, and the time intervals between packet arrivals. This filtering guarantees that the studied data originates solely from pertinent WebRTC sessions, excluding any unrelated network traffic.

4. QoS Calculation

Each Quality of Service (QoS) parameter in this study was derived from data collected by Wireshark during each test scenario. The examined characteristics encompassed throughput, packet loss, latency, and jitter. The four factors assess the quality of WebRTC-based video streaming, as each directly influences data transmission smoothness, network stability, and the quality of video material received by the viewer[21].

Throughput measures the actual data rate successfully transmitted during a certain observation period. The throughput value is determined by dividing the total data transmitted in bytes by the duration of the observation period. As the raw data from Wireshark is often represented in bytes, this value is multiplied by 8 to convert it to bits[22]. The formula used is as follows:

$$Throughput\ (bps) = \frac{Total\ Bytes\ \times 8}{Time\ Span}$$

$$Throughput\ (Mbps) = \frac{Total\ Bytes\ \times 8}{1.000.000}$$

Packet loss quantifies the percentage of packets that are lost during transmission. This indicator is determined by assessing the disparity between the quantity of packets transmitted and the quantity of packets received in relation to the total packets sent. A low packet loss rate signifies that the majority of packets were successfully received by the viewer, whereas a high packet loss rate may result in video playback complications, including frame loss, stuttering, freezing, or audio-video synchronization problems[23]. The formula used is as follows:

$$Packet\ Loss\ (\%) = \frac{Packets\ Sent - Packets\ Received}{Packet\ Sent} \times 100$$

Delay is utilized to quantify the time interval in packet arrival throughout the transmission process. This study calculates the delay value as the difference between the arrival times of the current packet and the preceding packet. Upon obtaining all delay values, the average delay is computed by dividing the total delay by the number of samples, thereafter converting it to milliseconds by multiplying by 1000. The employed formula is as follows:

$$Delay_i = Time_i - Time_{i-1}$$

$$Packet\ Loss\ (\%) = \frac{Packets\ Sent - Packet\ Receiver}{Packets\ Sent} \times 100$$

Jitter quantifies fluctuations in the latency between packets. This value is crucial in real-time communication, as packets arriving at irregular intervals might result in video stuttering or audio desynchronization. The jitter value is determined by the difference between the latency of the current packet and that of the preceding packet. The absolute value is employed to ensure that the delay variation is consistently computed as a positive quantity. The average jitter is computed by dividing the total jitter by the number of jitter samples and subsequently converted to milliseconds[24]. The formula used is as follows:

$$Jitter_i = |Delay_i - Delay_{i-1}|$$

$$Average\ Jitter\ (ms) = \frac{Total\ Jitter}{Jitter\ Samples} \times 1000$$

Employing these algorithms, each test scenario may be quantitatively assessed and objectively contrasted. Throughput calculations assess the system's capacity to transmit video data; packet loss measures the reliability of packet delivery; delay evaluates the responsiveness of the transmission; and jitter reflects the consistency of packet arrival times. The amalgamation of these four factors forms the foundation for assessing the efficacy of WebRTC-based video streaming systems with varying audience counts.

5. QoS Benchmarking Method

Following acquisition of the QoS values, the measurement outcomes were evaluated utilizing the QoS benchmarking methodology. This method was employed to evaluate the throughput, packet loss, delay, and jitter metrics in each scenario relative to network quality standards. The delay was assessed using the ITU-T G.114 standard, as this metric is closely associated with latency in real-time communication.[25]. Simultaneously, packet loss and jitter were assessed utilizing the TIPHON benchmark, as these metrics signify the stability of packet transmission in multimedia services. A comparative assessment was performed regarding video transmission requirements and the existing network capacity, as throughput demands are significantly influenced by resolution, bitrate, and viewer count[26].

Table 1. The QoS benchmarking criteria used in the evaluation

Parameters	Benchmark	Interpretation
Throughput	Evaluated based on bitrate adequacy and cross-scenario comparisons	The higher the throughput, the greater the system's ability to deliver video content to viewers.

Parameters	Benchmark	Interpretation
Packet Loss	TIPHON: The lower the packet loss percentage, the better the transmission quality	Low packet loss ensures the integrity of frames, audio, and video continuity.
Delay	ITU-T G.114: Low delay is required for interactive communication	High latency reduces the responsiveness of video conferences and interactive live streams.
Jitter	TIPHON: Low jitter indicates more stable packet arrival variations	Low jitter ensures smooth frame rates and audio-video synchronization.

RESULT AND DISCUSSION

Before presenting the QoS measurement findings, Figure 5 displays the test paperwork, which depicts the real experimental settings for each scenario. This guide shows how to set up a WebRTC-based video streaming system on a local network, with one broadcaster device as the media source and numerous viewer devices as stream recipients.

Figure 5. Documentation of WebRTC system testing on a local network

This documentation is supplied to reinforce the broad overview of the experimental arrangement, whilst the QoS analysis results are validated with quantitative data, specifically throughput, packet loss, delay, and jitter for each scenario. Table 2 summarizes the QoS measurement findings for all three test situations. To maintain compatibility with network units, Wireshark throughput measurements were changed from Kbps to Mbps. Thus, the values 2,524.69 Kbps, 4,322.85 Kbps, and 5,026.36 Kbps correspond to 2.525 Mbps, 4.323 Mbps, and 5.026 Mbps, respectively.

Table 2. Results of QoS parameter measurements in three WebRTC scenarios

QoS Parameters	Scenario 1	Scenario 2	Scenario 3
Throughput (Mbps)	2,525	4,323	5,026
Original throughput (Kbps)	2.524,69	4.322,85	5.026,3
Packet Loss (%)	0.002	0.0019	0,2
Total Delay (s)	26,31566	31,580887	33,509867
Average Delay (ms)	572,08	751,92	349,06
Total Jitter (s)	0,226086	0,086963	0,156299
Average Jitter (ms)	5,024	2,121	1,645

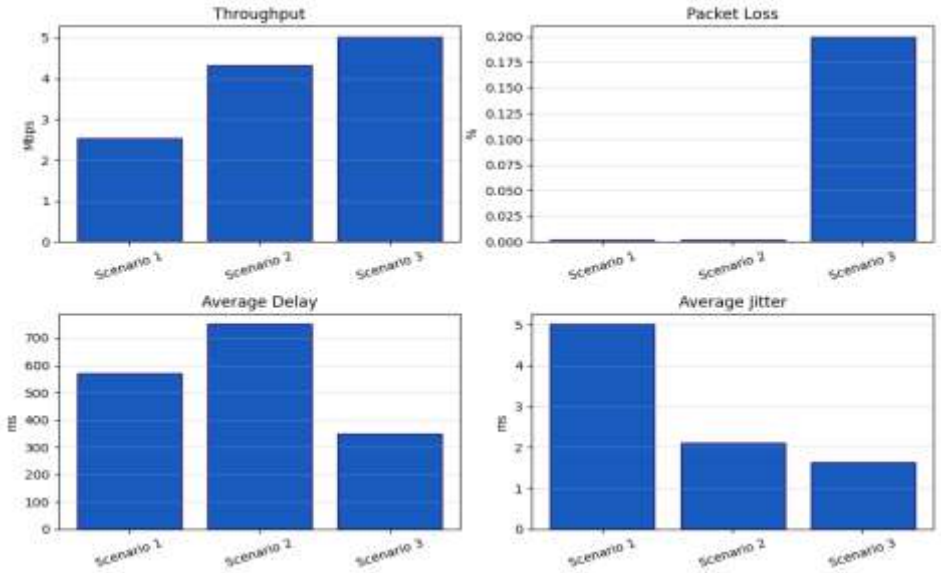

Figure 6. QoS Comparison Chart for Each Scenario

The test results reveal that as the number of viewers increases, so do the data transmission requirements. The throughput increased from 2,525 Mbps in Scenario 1 to 4,323 Mbps in Scenario 2 and 5,026 Mbps in Scenario 3. This increase implies that when more viewers see the stream, the system must process a higher volume of media data. In practice, faster throughput allows video footage to retain its visual quality, particularly when the information has a higher resolution or motion complexity. However, higher throughput necessitates adequate bandwidth to prevent video quality degradation owing to adaptive compression, frame dropouts, or packet delivery delays.

Packet loss in Scenarios 1 and 2 was extremely minimal, at 0.002% and 0.0019%, respectively. These numbers suggest that nearly all packets were correctly received, hence the likelihood of visual or auditory interference is low. In Scenario 3, packet loss rose to 0.2%. Although this score is still deemed modest, it shows that the rise in viewers is beginning to put strain on network stability. In the context of video material, increasing packet loss might result in dropped frames, visual glitches, temporary freezes, or audio-video synchronization difficulties. As a result, when a WebRTC system is intended to handle a large number of viewers, bandwidth control and connection reliability become crucial in ensuring that viewers receive high-quality material.

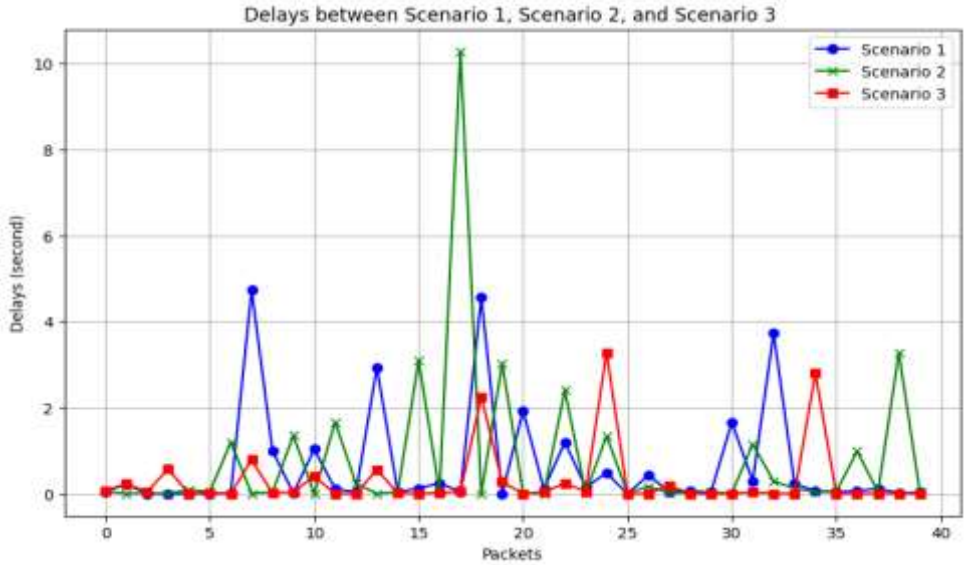

Figure 7. Comparison of Inter-Packet Delay Across Three WebRTC Scenarios

According to the delay graph in Figure 7, most packets in all three circumstances have low delay, however there are a few spikes in delay values at specific places. Scenario 1 exhibits many delay peaks, including those at 4.7 and 4.6 seconds, however the majority of packets are in the low range. Scenario 2 has the most dramatic delay spikes, topping 10 seconds around the 17th packet, as well as multiple other spikes that exceed 3 seconds. This suggests increased network oscillations or packet waiting in the case with two viewers. Scenario 3 has a more regulated delay pattern than Scenario 2, while multiple delay peaks of around 2-3 seconds still occur. Although total delay increases from Scenario 1 to Scenario 3, Scenario 3 has the lowest average delay at 349.06 ms. These findings demonstrate that additional viewers does not always result in a linear increase in average delay, since the delay distribution is influenced by channel stability, packet transmission patterns, and transient network conditions during testing.

The effect of delay on video content is substantial. High delay makes the video appear late to the viewer, particularly in interactive interactions like video conferences. In one-way live streaming, delay is bearable as long as the content remains constant; but, in two-way communication, large delay lowers conversation responsiveness. As a result, while WebRTC is capable of keeping most packets at a low delay, the presence of delay spikes in the graph suggests that the system still requires optimization to ensure that real-time video content remains responsive, particularly as the number of viewers grows.

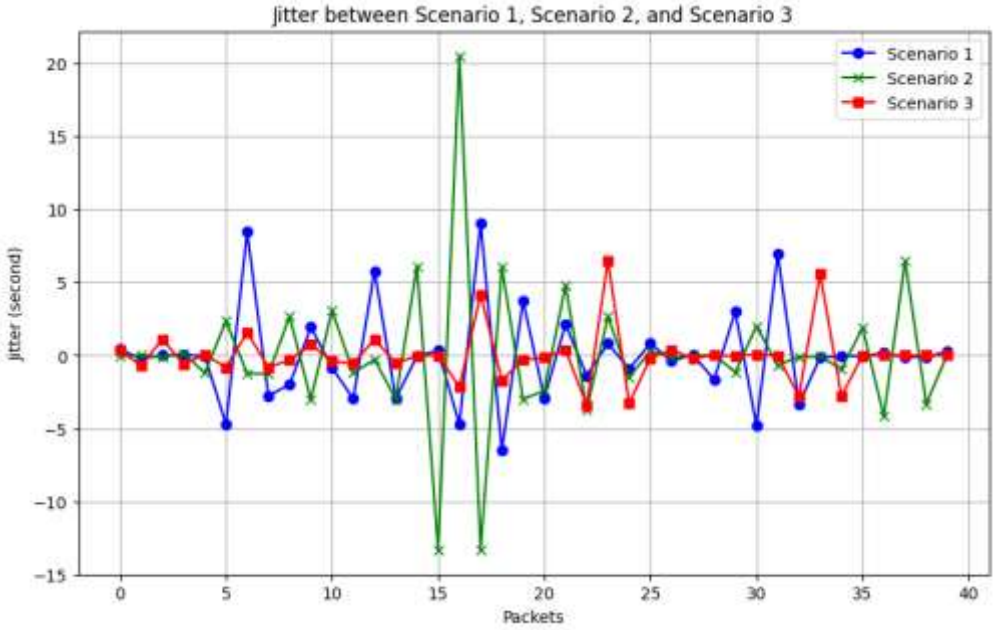

Figure 8. Comparison of Inter-Packet Jitter Across Three WebRTC Scenarios

The jitter graph in Figure 8 depicts the variations in packet arrival times for each situation. The graph's positive and negative values depict variations in inter-packet delay; a negative value implies that the next packet's delay is shorter than the previous one's, but negative jitter does not necessarily signal poor quality of service. Scenario 1 exhibits significant swings at various locations, with a peak reaching 9 seconds and a trough around -6.5 seconds. Scenario 2 has the most dramatic oscillations, with a high of more than 20 seconds and a decline to around -13 seconds between the 15th and 18th packets. This is consistent with the delay graph, which also shows a significant rise in Scenario 2. Scenario 3 has more modest oscillations than Scenario 2, but there are still some spikes at specific periods.

Scenario 3 had the lowest average jitter (1.645 ms), followed by Scenario 2 (2.121 ms) and Scenario 1 (5.024 ms). A low average jitter value suggests that most packets are received in reasonably consistent time intervals. In the context of video content, minimal jitter helps to ensure smooth playing and audio-video synchronization. In contrast, severe jitter can cause stuttering,

choppy audio, and frame inconsistencies. As a result, despite having the most viewers and the worst packet loss, Scenario 3's average jitter value demonstrates that the system can still maintain general stability in the face of packet arrival changes.

Overall, the study found that WebRTC can support local streaming with up to three devices, but each additional viewer has ramifications for throughput needs and the possibility of higher packet loss. QoS results represent not just the network's technical state, but also the quality of the received video content. Sufficient throughput guarantees that the system can provide video material at the desired quality, while minimal packet loss preserves frame integrity, low delay ensures responsiveness, and low jitter ensures a pleasant viewing experience. Thus, the WebRTC broadcaster-viewer system on a local network can serve as a foundation for real-time communication, but larger-scale implementations necessitate network optimization, bitrate adjustment, and continual QoS monitoring.

CONCLUSION

This study assesses the Quality of Service performance of a WebRTC-based video streaming system within a local network, using three distinct viewer counts. The findings indicate that throughput escalates with the rise in viewer count, attaining 2.525 Mbps in Scenario 1, 4.323 Mbps in Scenario 2, and 5.026 Mbps in Scenario 3. Packet loss was minimal, albeit it rose to 0.2% in Scenario 3. Scenario 3 exhibited the lowest average delay of 349.06 ms, whilst Scenario 2 recorded the highest average delay of 751.92 ms. Scenario 3 exhibited the most favorable average jitter of 1.645 ms. The findings demonstrate that WebRTC can provide multi-viewer streaming over local networks; however, a rise in viewer count necessitates management of bandwidth, bitrate, and network stability to maintain optimal video quality for viewers. Additional research is advised to evaluate bigger cohorts of viewers, differences in video resolution, adaptive bitrate, and Quality of Experience (QoE) assessments based on user perception.

REFERENCES

[1] H. Mahmoud and R. Abozariba, “A systematic review on WebRTC for potential applications and challenges beyond audio video streaming,” *Multimed Tools Appl*, vol. 84, no. 6, pp. 2909–2946, Nov. 2024, doi: 10.1007/s11042-024-20448-9.

[2] G. Gorrochategui, U. Zulaika, and P. Garaizar, “STUN comprehension-optional attributes as a covert channel: Design, implementation, and detection,” *Computers & Security*, vol. 170, p. 105043, Nov. 2026, doi: 10.1016/j.cose.2026.105043.

[3] W.-C. Shih *et al.*, “The Construction of a Stream Service Application with DeepStream and Simple Realtime Server Using Containerization for Edge Computing,” *Sensors*, vol. 25, no. 1, p. 259, Jan. 2025, doi: 10.3390/s25010259.

[4] S. Otto, L. B. Bertel, N. E. R. Lyngdorf, A. O. Markman, T. Andersen, and T. Ryberg, “Emerging Digital Practices Supporting Student-Centered Learning Environments in Higher Education: A Review of Literature and Lessons Learned from the Covid-19 Pandemic,” *Educ Inf Technol*, vol. 29, no. 2, pp. 1673–1696, Feb. 2024, doi: 10.1007/s10639-023-11789-3.

[5] A. Aytekin, H. Özköse, F. Akgün, and A. Ayaz, “The Role of Digital Transformation in Local Governments During the COVID-19 Pandemic in Türkiye: Opportunities and Challenges,” *J Knowl Econ*, vol. 17, no. 1, pp. 1503–1537, Mar. 2025, doi: 10.1007/s13132-025-02651-7.

[6] D. Diaz, D. Stolarz, and J. Aguerre, “Toward Real-Time Video Streaming Over WebRTC Data Channels to Support Supplementary Video Codecs and Formats in the Browser,” in *2024 IEEE International Conference and Expo on Real Time Communications at IIT (RTC)*, Chicago, IL, USA: IEEE, Oct. 2024, pp. 39–45. doi: 10.1109/RTC62204.2024.10739162.

[7] S. Vogel, F. Wege, and A. Monti, *WebRTC-based plug-&-play signal transport for peer-to-peer connectivity between DRTSs, IEDs and operators*. DE: VDE VERLAG GMBH, 2025. doi: 10.30420/566464017.
[8] F. M. García, S. Schez-Sobrino, C. Glez-Morcillo, J. J. Castro-Schez, J. A. Albusac, and D. Vallejo, "RTC-MR: A WebRTC-based framework for real-time communication in Mixed Reality," *Software Impacts*, vol. 23, p. 100727, Mar. 2025, doi: 10.1016/j.simpa.2024.100727.
[9] S. A. Mahmood, N. M. Edan, and M. Kherallah, "Improving WebRTC Quality-of-Service Using SDN and a Load Balancing Strategy," in *2024 25th International Arab Conference on Information Technology (ACIT)*, Zarqa, Jordan: IEEE, Dec. 2024, pp. 1–9. doi: 10.1109/ACIT62805.2024.10877267.
[10] D. Markudova and M. Meo, "Advancing Congestion Control for Real-Time Communications With Reinforcement Learning: The ReCoCo Framework," *IEEE Trans. Netw. Serv. Manage.*, vol. 23, pp. 3998–4008, 2026, doi: 10.1109/TNSM.2026.3683265.
[11] H. R. Oktaseli and A. A. Slameto, "Evaluation of Wireless LAN Quality of Service (QoS) in Primary Education Using TIPHON Standards," *JAIC*, vol. 9, no. 2, pp. 393–403, Mar. 2025, doi: 10.30871/jaic.v9i2.8979.
[12] M. S. R, E. Adriono, and Y. E. Windarto, "Web-Native Architecture for Real-Time ROV Video Streaming and Analytics Next.js, WebRTC, and Python FastAPI," in *2025 2nd Beyond Technology Summit on Informatics International Conference (BTS-I2C)*, Jember, East Java, Indonesia: IEEE, Dec. 2025, pp. 404–409. doi: 10.1109/BTS-I2C67944.2025.11399354.
[13] J. Nakazato, K. Nakagawa, K. Itoh, R. Fontugne, M. Tsukada, and H. Esaki, "WebRTC over 5 G: A Study of Remote Collaboration QoS in Mobile Environment," *J Netw Syst Manage*, vol. 32, no. 1, p. 1, Jan. 2024, doi: 10.1007/s10922-023-09778-5.
[14] P. Bhalange, S. Thakur, and M. Chen, "Peer-to-Peer Synchronization for Collaborative Media Streaming," in *2025 IEEE 8th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, San Jose, CA, USA: IEEE, Aug. 2025, pp. 213–218. doi: 10.1109/MIPR67560.2025.00041.
[15] X. Li, X. Tian, and Z. Yang, "A Web-Based Lightweight Peer-to-Peer Multi-Endpoint Remote Control System," in *2025 10th International Conference on Computer and Communication System (ICCCS)*, Chengdu, China: IEEE, Apr. 2025, pp. 1045–1053. doi: 10.1109/ICCCS65393.2025.11069457.
[16] V. K. Singh, P. Awasthi, L. Tripathi, S. Thakur, and P. Singh, "Mern (Mongodb, Express-Js, React-Js, Node-Js) Stack Web-Based Streaming and Broadcasting Application," in *2024 International Conference on Electrical Electronics and Computing Technologies (ICEECT)*, Greater Noida, India: IEEE, Aug. 2024, pp. 1–6. doi: 10.1109/ICEECT61758.2024.10738869.
[17] M. Nagy, T. Lévai, F. Németh, A. Panda, G. Antichi, and G. Rétvári, "Elastic Scaling of Real-Time Communication Services," *IEEE Trans. Netw. Serv. Manage.*, vol. 23, pp. 3393–3405, 2026, doi: 10.1109/TNSM.2026.3674598.
[18] R. Belda, J. Llacer, P. Arce, and J. C. Guerri, "SwarmLayer: A Hybrid P2P Assisted Solution for MPEG-DASH Streaming," in *2025 International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, Barcelona, Spain: IEEE, Oct. 2025, pp. 507–514. doi: 10.1109/MSWiM67937.2025.11309056.
[19] X. Tang *et al.*, "Low-Latency Video Streaming: Applications, Challenges, and Trends," *IEEE Commun. Surv. Tutorials*, vol. 28, pp. 6384–6414, 2026, doi: 10.1109/COMST.2026.3689266.
[20] N. Smirnov and S. Tomforde, "Real-time rate control of WebRTC video streams in 5G networks: Improving quality of experience with Deep Reinforcement Learning," *Journal of Systems Architecture*, vol. 148, p. 103066, Mar. 2024, doi: 10.1016/j.sysarc.2024.103066.
[21] I. Chicano-Capelo, F. Gortázar, and M. Gallego, "Quality of Experience Under Huge Load for WebRTC Applications: A Case Study of Three Media Servers," *IEEE Access*, vol. 13, pp. 140440–140461, 2025, doi: 10.1109/ACCESS.2025.3589785.

[22] D. Chmieliauskas and Š. Paulikas, "Evaluation of Uplink Video Streaming QoE in 4G and 5G Cellular Networks Using Real-World Measurements," *IEEE Access*, vol. 13, pp. 53996–54018, 2025, doi: 10.1109/ACCESS.2025.3554340.

[23] A. Armendariz, J. Joskowicz, R. Sotelo, and M. Liu, "A Test Bed for Subjective Multimedia Quality Evaluation in Videoconferencing Systems," in *2024 IEEE International Conference on Consumer Electronics (ICCE)*, Las Vegas, NV, USA: IEEE, Jan. 2024, pp. 1–4. doi: 10.1109/ICCE59016.2024.10444493.

[24] T. S, B. V. Pavani, C. Divya, D. Mahitha, and G. Thanmai, "Enhancing VoIP in Telemedicine by Hybrid Jitter Buffer Algorithm," in *2024 Control Instrumentation System Conference (CISCON)*, Manipal, India: IEEE, Aug. 2024, pp. 1–6. doi: 10.1109/CISCON62171.2024.10696234.

[25] M. N. Wirawan, M. Lubis, and M. T. Kurniawan, "Evaluating Quality of Service: Throughput, Packet Loss, and Delay in Tree Topology with Ryu and Pox Controllers in Software-Defined Network," in *2024 4th International Conference of Science and Information Technology in Smart Administration (ICSINTESA)*, Balikpapan, Indonesia: IEEE, Jul. 2024, pp. 457–462. doi: 10.1109/ICSINTESA62455.2024.10748026.

[26] A. B. Grossi, R. I. Tavares Da Costa Filho, and L. P. Gaspary, "On the quality of WebRTC-based videoconferencing under adverse and mobility scenarios," *Ann. Telecommun.*, vol. 80, no. 9–10, pp. 851–866, Oct. 2025, doi: 10.1007/s12243-025-01102-3.