WebRTC-Based Volumetric Video Conferencing: SFU Architecture Evaluation and Benchmarking

Anonymous Author(s)

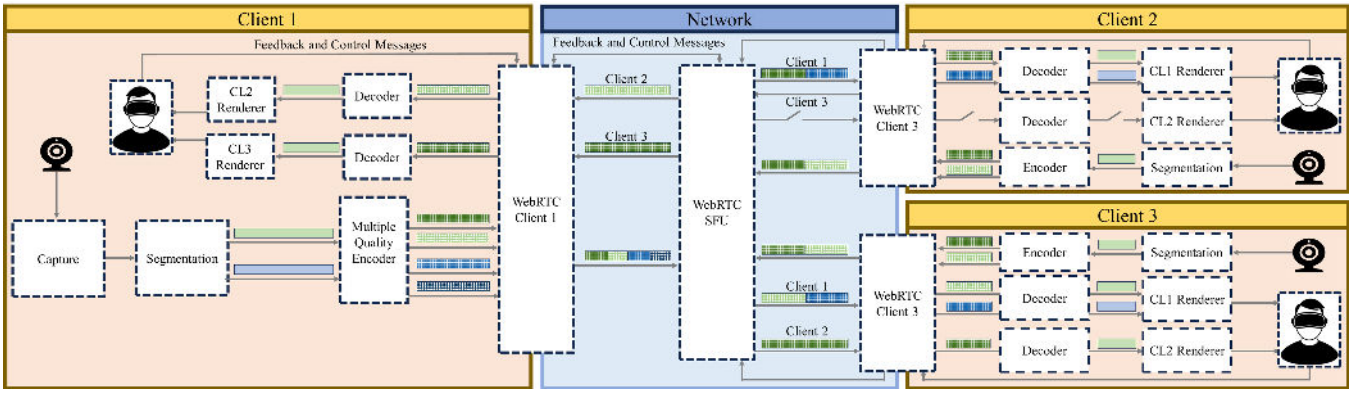


Figure 1: The proposed codec-independent WebRTC SFU enables selective forwarding of quality levels per client.

Abstract

Immersive technologies promise to revolutionize remote communication through enhanced sense of presence and interactivity. To enable interactive experiences, reliable low-latency transport mechanisms are needed to handle the large volumes of data created by complex 3D objects and scenes. In this paper, we propose an open-source, codec-independent, selective forwarding unit (SFU) for real-time volumetric video streaming using WebRTC. For evaluation purposes, we provide a reference client implementation by extending VR2Gather, a TCP-based system for immersive telecommunication. We conduct extensive evaluations using both new and existing datasets to compare the performance of WebRTC against the existing TCP-based protocols under diverse configurations in an emulated testbed environment. The evaluations demonstrate that WebRTC outperforms other protocols in high-latency scenarios and adapts video quality to user movement 13% and 36% faster than its TCP-based counterparts in networks with 5 ms and 10 ms of network latency, respectively.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

NOSSDAV'26, April 4-8, 2026, Hong Kong SAR

© 2026 ACM.

ACM ISBN TODO
https://doi.org/TODO

CCS Concepts

• **Information systems** → **Multimedia streaming**; • **Human-centered computing** → *Virtual reality*.

Keywords

Volumetric Video, Conferencing, Virtual Reality, WebRTC

ACM Reference Format:
Anonymous Author(s). 2026. WebRTC-Based Volumetric Video Conferencing: SFU Architecture Evaluation and Benchmarking. In *The 36th Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'26), April 4-8, 2026, Hong Kong SAR.* ACM, New York, NY, USA, 7 pages. https://doi.org/TODO

1 Introduction

Immersive technologies are redefining remote communication by enhancing co-presence and interactivity, making virtual exchanges increasingly similar to face-to-face interactions [15]. Transmission protocols play a key role in enabling real-time volumetric video communication, allowing the choice between reliability when using TCP, or achieving the lowest latency with UDP [2]. The UDP-based WebRTC is commonly used to provide low-latency streaming, with optional reliability mechanisms [8]. Although WebRTC's peer-to-peer model enables streaming, scalability issues arise as the number of users grows, motivating centralized solutions such as selective forwarding units (SFUs). Despite this recent focus on volumetric video, to the best of our knowledge, no prior work has directly compared WebRTC against TCP-based solutions within the same reference software, as the two are seldom implemented together [26].

To address these gaps, we propose a codec-independent, open-source [20] WebRTC-based SFU for real-time many-to-many volumetric streaming. As shown in Figure 1, clients capture, segment, and encode volumetric data at multiple qualities, which the SFU selectively forwards based on network and user conditions. Receivers then decode and render tracks in real time, supported by feedback messages for adaptive quality and synchronization. The proposed solution can be integrated into volumetric streaming platforms. For evaluation purposes, a reference client implementation is provided based on VR2Gather [12], providing a reference for benchmarking UDP- versus TCP-based (direct TCP, SocketIO, and LL-DASH) protocols in immersive communication. In summary, the main contributions of this paper are as follows:

- A codec-independent WebRTC-based SFU for volumetric conferencing with adaptive quality;
- An evaluation of the system with multiple network protocols, quality schemes, and network latency;
- A reference WebRTC client implementation on how the solution can be deployed in different platforms;
- A realistic bidirectional dataset of two users.

2 Related Work

This section presents volumetric conferencing systems, reviews related work on point cloud codecs, and discusses adaptive streaming with existing codecs.

Volumetric Video Conferencing Systems. Many immersive platforms have been designed to facilitate volumetric conferencing. High-fidelity solutions provide optimal user quality, but require complex setups, making them less appealing [14]. In contrast, more scalable systems have used lower quality representations with multipoint control unit (MCU)-based scalability [11] to improve system performance. In [5], performance is further improved by using quality adaptation based on the network conditions and field of view (FoV) to reduce latency and improve scalability. Finally, VR2Gather [12, 26] offers a modular platform for volumetric communication, supporting various reliable transport protocols, but lacks UDP-based solutions. We therefore adopt VR2Gather to evaluate our WebRTC implementation, enabling benchmarking of UDP- versus TCP-based transmission.

Point Cloud Compression. The high bitrates of uncompressed point clouds demand high levels of compression [24]. Non-real-time codecs, such as video point cloud compression (V-PCC), achieve low bitrates similar to 2D video but at high computational cost [9]. In contrast, Draco [7] focuses on real-time encoding at the cost of having bitrates several times higher than V-PCC. The cwipc codec extends the MPEG anchor codec [16], storing geometry in an octree and encoding color via JPEG compression. The codec offers a strong balance between efficiency and speed, achieving real-time performance at 15 FPS with point clouds containing 200.000 points, making it well suited for live volumetric streaming.

Adaptive Streaming of Immersive Video. Because volumetric video imposes heavy bandwidth and computational demands, several solutions have focused on optimizing transmission for end users [27]. Traditional adaptive streaming has been extended to account for interactivity, selecting representations based on viewing direction and network conditions [18, 28]. Another strategy segments content into tiles encoded at multiple qualities and delivered via DASH, allowing clients to request appropriate quality levels [22, 23, 25]. Codecs like cwipc inherently support multi-quality tiling, while other approaches use codec-independent multiple description coding (MDC)-based methods to combine quality layers [5]. In this paper, we provide a codec-independent solution for delivering volumetric video using WebRTC to achieve a lower end-to-end delay and provide a means towards quality adaptation based on network conditions.

3 Architectural Design and Implementation

In this section, we outline the core concepts of the WebRTC protocol suite, present the high-level design decisions of our codec-independent SFU and describe its implementation.

WebRTC Background. WebRTC is commonly used for its low latency and potential scalability with server-based architectures. MCU-based systems merge all incoming streams into a single stream per client, but this aggregation is computationally expensive and memory-intensive, limiting scalability and increasing latency due to the need to wait for all inputs [10]. In contrast, SFU-based architectures forward packets directly and immediately to relevant clients, reducing latency and memory usage, and enabling better packet pacing, which improves overall network flow quality and decreases the likelihood of packet loss. WebRTC establishes peer connections through the exchange of session description protocol (SDP) messages, which describe session parameters. Using an offer–answer model, one peer sends an SDP offer with supported settings, while the other peer responds with an SDP answer selecting compatible parameters [21].

Architecture. High-performance many-to-many volumetric conferencing requires a mediated architecture. To avoid the computational cost and latency of MCU-based designs, the proposed system adopts an SFU architecture. As mentioned in Section 2, adaptive streaming reduces network load and improves quality. WebRTC's native support for multiple tracks enables the SFU to transmit data over distinct streams, allowing point clouds to be partitioned via tiling or semantic segmentation and delivered as separate tracks.

Furthermore, multiple parallel encoders can be used to generate several quality levels, each transmitted over a separate track. As these tracks represent the same content, the SFU forwards at most one per client, selecting the appropriate quality based on bandwidth, user context, and client feedback. Although each object has a transmission track for each quality, each receiving client maintains only a single track per object. This mechanism minimizes open tracks and computational load, while allowing the SFU to switch quality by reassigning the sender on the existing forwarding track, without needing SDP renegotiation. Together, these mechanisms support diverse coding strategies, including segmentation-based schemes with multiple qualities per segment and MDC approaches that combine segments to improve quality.

SFU Implementation. The SFU is implemented by adapting Golang Pion [17], which provides the full WebRTC protocol suite and server-side bandwidth estimation via Google congestion control (GCC). A custom media track handles the transcoding and packetizing of the point-cloud frames through a dedicated payloader, while audio is sent via a separate track. Negative acknowledgment (NACK)-based transmissions are used to ensure frames are decodable in the event of packet loss. Combined with GCC, this approach has shown strong performance for volumetric video streaming in congested networks [4]. Upon connection, the server signals the tracks of the new client to the other peers, prompting them to instantiate the required tracks for transmission. The server also supports quality selection and enables the support of additional SFU control messages with minimal overhead. To enable seamless quality switching, a custom packet interceptor preserves sequence numbering across quality tracks. Without this interceptor, sequence gaps would be interpreted as packet loss, triggering unnecessary retransmissions of the wrong quality track. The use of an optional session traversal utilities for NAT (STUN) server mitigates problems in networks that employ network address translation (NAT) [6].

WebRTC Peer Reference Implementation. Any WebRTC client library that supports media tracks of non-standardized codecs can be used to connect to the proposed SFU. However, to fully leverage the capabilities of the SFU, the client must also support NACK, GCC, and the associated feedback messages. Following a comprehensive evaluation of available WebRTC libraries using the criteria mentioned above, the Pion library was selected for the reference client implementation. However, the Golang language is not natively compatible with Unity, the game engine employed by VR2Gather. To bridge this gap, a library was developed to write outgoing frames to a local UDP port, which is continuously polled by the Golang client. The additional overhead introduced by this approach was measured at approximately 1 ms, a negligible cost given the performance benefits it enables.

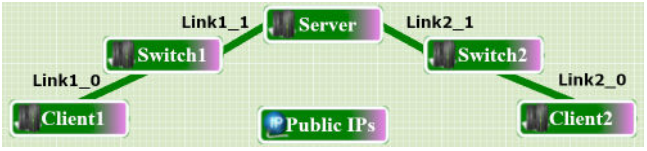


Figure 2: Experimental setup on [Anon] testbed [1].

4 Evaluation Methodology and Setup

In this section, we first discuss the evaluation methodology, outlining the experiments that were performed. Then, the experimental setup and the considered metrics are presented.

4.1 Evaluation Methodology

VR2Gather currently supports four protocols: Direct TCP, TCP Reflector, SocketIO, and the proposed WebRTC implementation. Each protocol allows the following modules to be enabled: compression, multi-quality encoding and tiling.

For unidirectional scenarios, a single sender, responsible for playback and encoding, and a single receiver, responsible for decoding and rendering, are considered. The objective is to evaluate the efficiency of the protocols in the VR2Gather platform, while assessing the impact of encoding frame rate, quality level, and tiling on the performance of the system.

For interactive scenarios, the experiments compare quality adaptation latency when using WebRTC or Direct TCP, the only protocols in VR2Gather that support quality adaptation at acceptable performance. In this bidirectional setup, both clients replay recorded content and use captured movement to drive quality adaptation decisions [19]. Each client operates at a different frame rate to assess the impact of the sending frame rate on the other client's quality adaptation latency. To enable fair comparison with prior VR2Gather-based work [13], the existing decision-making logic of VR2Gather is used and quality decisions are exchanged via a websocket, rather than relying on a dedicated bandwidth estimator.

4.2 Experimental Setup

Figure 2 shows the considered topology in which two clients are connected to a centralized server through custom switches. The involved machines are interconnected using a 1 Gb/s link, and each have one of the following specifications:

- CPU: Intel Xeon E3-1220 v3, RAM: 16 GB DDR3
- CPU: Intel Xeon E3-5645 v3, RAM: 24 GB DDR3

Because these nodes lack the required resources for rendering, higher-tier laptops are directly connected to the client nodes using VPN. This setup allows the laptops to be forced to take a specific route throughout the network, which ensures that any network impairments that are applied on traffic going through the network, is also applied during the experiments. All machines are located within the same network, and thus introduce sub 1 ms latency, as well as being synchronized with the network time protocol (NTP).

Table 1: System performance in terms of latency, bandwidth, frame rate and CPU usage for sessions with two participants that exchange the *Loot* object without additional network latency.

Protocol	Tiles	Octree level	Target FPS	Latency (ms)	Bandwidth (Mb/s)	Obtained FPS	CPU
Direct TCP	1	7	15	45.29 ± 4.15	3.70 ± 0.37	14.90 ± 0.42	39.88 ± 8.27
Direct TCP	1	9	15	334.20 ± 25.66	11.17 ± 1.68	7.18 ± 0.86	44.32 ± 19.63
TCP Reflector	1	7	15	46.11 ± 2.73	3.75 ± 0.34	14.89 ± 0.40	39.95 ± 8.05
TCP Reflector	1	9	15	332.01 ± 26.88	11.36 ± 1.64	7.24 ± 0.86	43.64 ± 26.25
SocketIO	1	7	15	50.46 ± 3.48	3.69 ± 0.37	14.98 ± 0.64	42.47 ± 17.67
SocketIO	1	9	15	337.40 ± 32.90	11.33 ± 1.45	7.59 ± 0.78	45.10 ± 26.14
WebRTC	1	7	15	43.75 ± 3.19	3.94 ± 0.38	14.89 ± 0.46	40.69 ± 8.21
WebRTC	1	9	15	334.07 ± 27.54	12.28 ± 1.63	7.51 ± 0.83	45.89 ± 23.86

The following hardware is used:

- Client 1: CPU: Intel Core i7-12700H (2.30 GHz), RAM: 32 GB DDR5, GPU: NVIDIA GeForce RTX 3070 Ti
- Client 2: CPU: Intel Core i7-10750H (2.60 GHz), RAM: 16 GB DDR4, GPU: NVIDIA GeForce RTX 2070

To emulate wide-area network conditions, bidirectional latency between 0 and 50 ms is applied on the switch nodes, corresponding to half the client–server round-trip time (RTT).

For unidirectional experiments, the Loot sequence from the 8i dataset [3] at 15 and 30 FPS is considered. Point clouds are pre-scaled with a voxel size of 2.5 to reach a point count of 200,000 points same as in [26], and are split into four tiles as described in [22]. The MPEG anchor codec [16] is used to enable real-time encoding, with two quality levels (low quality=octree 7 and high quality=octree 9). In these experiments, client 1 is the sender, while client 2 is the receiver.

For interactive experiments, content was captured using two machines using four Microsoft Azure Kinect cameras, resulting in fifteen tiles. To invoke enough quality changes, two users dance together and rotate around each other during a one-minute session. To demonstrate platform flexibility, one user is captured at 15 FPS and the other at 30 FPS, with both clients acting as sender and receiver. As both encoding and decoding occur on each machine, reduced quality levels are used (low quality=octree 5 and high quality=octree 7).

4.3 Evaluation Metrics

To accurately evaluate the performance of the system, each experiment is performed with 10 iterations, each lasting about 1 minute. We consider the following metrics (with 95% confidence intervals): *pipeline latency, bandwidth and CPU usage, receiver FPS, quality change latency (time between requesting a quality change and the quality being rendered).*

5 Results and Discussion

In this section, the results are discussed. First, the results of the unidirectional scenario, along with an examination of the impact of tiling and network latency. Second, the results of the bidirectional scenario, focusing om the real-time capabilities of quality switching.

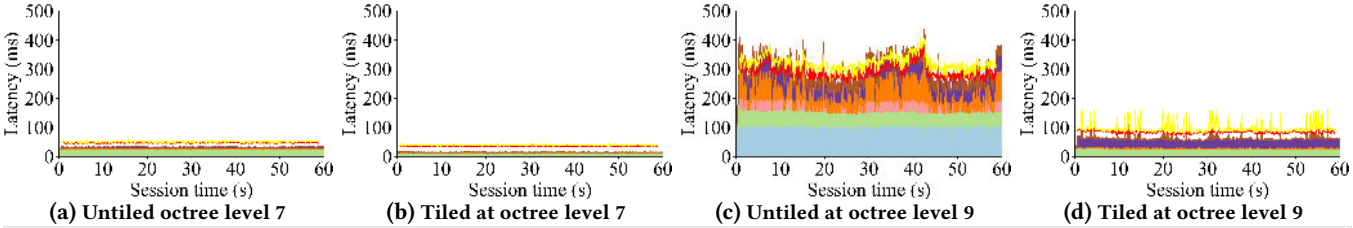
5.1 Unidirectional Content Benchmark

Impact of Protocols and Quality. Table 1 presents the results of the experiment with untiled content at 15 FPS for each protocol. All protocols exhibit latency values within a 5 ms margin. Additionally, during each iteration the latency is stable, with minor fluctuations attributable to non-transport components. WebRTC exhibits slightly higher bandwidth. This increased bandwidth stems from the larger frames, which lead to more packets, each incurring additional WebRTC header overhead. Additionally, Table 1 reveals that streaming at higher quality (octree level 9), prevents the system from maintaining the target frame rate of 15 FPS with the hardware used in the experiments. To alleviate this issue, VR2Gather enables tiling to improve encoding.

Impact of Tiling and Frame Rate. As shown in Table 2, tiling has a significant impact on the latency of the system, at the cost of increasing the overall bandwidth. Due to the parallel coding of each tile, encoding and decoding times are reduced by a factor of 4.63 and 3.47, respectively, making it possible to achieve higher frame rates and transmit higher-quality content at acceptable frame rates. However, there is an exception to this: when using an octree depth of 7 at 30 FPS, no change in latency is achieved. Although the coding time decreases, VR2Gather internally buffers the frames to maintain a stable FPS, resulting in no decrease in latency. As shown in Figure 3, the largest decrease in latency can be seen for higher octree levels, which shows that tiling removes any latency caused by the encoder queue. In effect, each frame is encoded and sent immediately upon production, maintaining the target rate of 15 FPS. This improvement enables the sender and receiver to sustain 15 FPS for an octree depth of 9. However, when increasing the frame rate to 30 FPS, the sender only achieves a send rate of 18 FPS compared to the captured rate of 30 FPS. For untiled content, with an octree depth of 9 and WebRTC, Table 2 shows that latency values are lower at a capture rate of 30 FPS compared to 15 FPS, despite both achieving the same rendering frame rate. This behavior occurs because of the limited capacity of the encoder queue. When a frame is added to a full queue, the oldest frame that is not currently being encoded is discarded. At higher capture

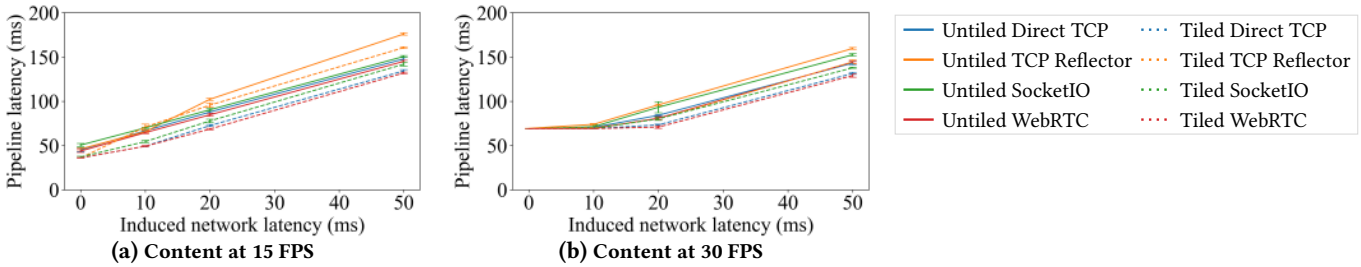
Table 2: System performance in terms of latency, bandwidth, frame rate and CPU usage for sessions with two participants that exchange the *Loot* object through WebRTC, for different video quality configurations.

Tiles	Octree level	Target FPS	Latency (ms)	Bandwidth (Mb/s)	Obtained FPS	CPU
1	7	15	43.75 ± 3.19	3.94 ± 0.38	14.89 ± 0.46	40.69 ± 8.21
4	7	15	36.02 ± 1.86	8.09 ± 0.67	14.90 ± 0.45	42.86 ± 8.16
1	9	15	334.07 ± 27.54	12.28 ± 1.63	7.51 ± 0.83	45.89 ± 23.86
4	9	15	111.07 ± 9.42	25.73 ± 2.79	14.99 ± 0.99	49.99 ± 19.01
1	7	30	69.14 ± 1.58	7.63 ± 0.72	30.07 ± 0.85	51.69 ± 8.84
4	7	30	69.99 ± 1.74	15.71 ± 1.52	30.04 ± 0.83	52.69 ± 7.41
1	9	30	289.48 ± 16.42	12.05 ± 1.54	7.38 ± 0.60	38.29 ± 19.66
4	9	30	161.68 ± 14.81	31.47 ± 4.18	18.77 ± 2.41	66.51 ± 30.56



(a) Untiled octree level 7 **(b) Tiled at octree level 7** **(c) Untiled at octree level 9** **(d) Tiled at octree level 9**

Figure 3: Latency contributions for different video quality configurations, for sessions with two participants and a target frame rate of 15 FPS using WebRTC.



(a) Content at 15 FPS **(b) Content at 30 FPS**

Figure 4: Latency contributions for different video quality configurations and induced network latency values, for sessions with two participants at 15 FPS and 30 FPS, using an octree depth of 7.

rates, frames enter the queue more frequently, leading to more frequent discarding. Consequently, the frames that are not dropped spend 56% less time in the encoder at a target FPS of 30 compared to 15 FPS. This demonstrates that, in certain scenarios, using a higher capture rate can be advantageous even if the client cannot encode frames at the same rate.

Impact of Network Latency. Figure 4 shows the impact of network latency. The results indicate that most of the protocols achieve similar results, with one exception at high latencies: the TCP Reflector. In case of 50 ms of added latency, this protocol exhibits a latency that is 21% higher than that of WebRTC. Furthermore, for tiled content at 15 FPS, the TCP Reflector's latency is larger than the latency of the other protocols with no tiling. This disparity becomes less pronounced with content at 30 FP, where the latency of the TCP Reflector remains 10% higher than WebRTC but still outperforms the untiled performance of the other protocols. Similar to the results in Table 2, all protocols show no reduction in latency when tiling is used at a capture rate of 30 FPS in the absence of induced network delay. When a small amount of latency is introduced, the overall delay remains unaffected because of buffering in VR2Gather's encoding queue. However, once the induced network latency reaches 20 ms, tiling leads to a measurable reduction in overall latency. Under these conditions, Direct TCP and WebRTC outperform the other two protocols, achieving comparable latency without tiling to what the remaining protocols obtain with tiling.

5.2 Quality Decision Latency for Bidirectional Interactive Sessions

Tiling is also used for quality adaptation. In VR2Gather, this is achieved by assigning it the closest tile with the highest quality, while allocating lower quality to the remaining tiles. This is particularly crucial in scenarios with a lot movement, like the dance in these experiments. Figure 5 shows the impact of quality changes when introducing a network latency of 10 ms. These quality changes are not immediately visible, potentially reducing the overall experience [23]. Consequently,

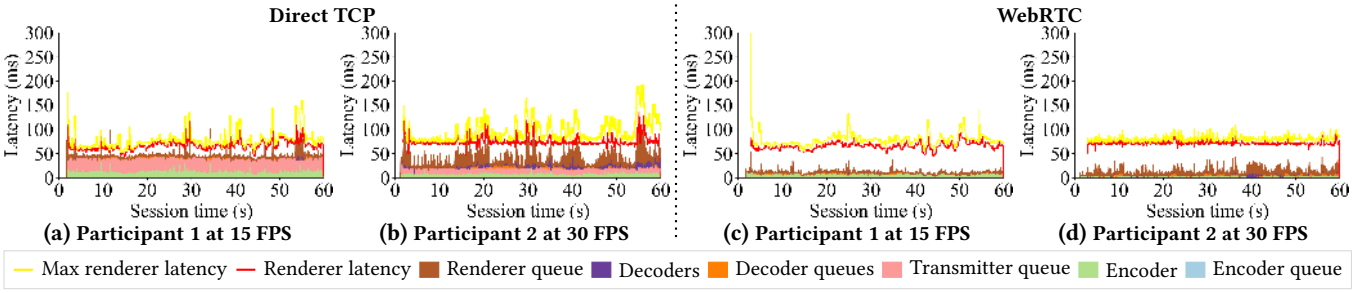


Figure 5: Latency contributions for Direct TCP and WebRTC with quality adaptation, using the captured dance sequence with the first quality at octree depth 7 and the second at depth 9 with 10 ms of added latency.

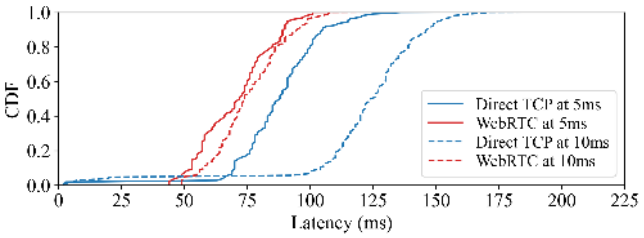


Figure 6: Time between the quality decision being made and the new chosen quality being visible.

the time required for a quality change to become visible serves as a valuable indicator of overall user experience. With Direct TCP, frequent and significant latency spikes occur during quality changes, a behavior not observed in for WebRTC. This difference is attributed to the distinct quality change mechanisms of the two protocols. In the WebRTC-based implementation, each tile is associated with a separate WebRTC media track. On the receiver side, a single track is created for each tile, allowing the receiver to simply signal the desired quality to the SFU, which then swaps the sender track being forwarded to the receiver track. This approach minimizes overhead during quality changes, reducing the likelihood of latency spikes. In contrast, the Direct TCP implementation in VR2Gather requires closing and reopening the socket for a tile whenever a quality change occurs, leading to the observed spikes in latency. This process involves multiple round-trips to transmit the new quality preferences, resulting in significant latency spikes until the connection is fully established. However, the advantage of the Direct TCP implementation is its efficient bandwidth usage, as only the quality levels requested by the receiver are transmitted. In contrast, the WebRTC implementation always sends every tile at all quality levels to the SFU, which then determines the appropriate quality to forward to the receivers. As the current VR2Gather adaptation algorithm only selects one tile to be the highest quality, the bandwidth used to transmit the 14 other high-quality tiles is wasted. However, as the WebRTC solution always forwards every quality to the SFU, the response time to implement the quality change is significantly lower. Because WebRTC requires only half an RTT to signal quality changes, adjustments can often be applied before the next frame is produced. This behavior is evident in Figure 6, where the choice latency remains nearly identical for both 5 ms and 10 ms delays when using WebRTC. In contrast, Direct TCP requires multiple RTTs to implement a quality change, significantly increasing latency. Even with a delay of 5 ms, Direct TCP cannot establish a new connection before the next frame. Additionally, both protocols perform significantly better compared to the publicly available LL-DASH results of VR2Gather [13]. In LL-DASH, quality adaptation occurs at the segment rather than per frame, making the latency dependent on segment size. While reducing the size can mitigate this issue, achieving performance comparable to WebRTC would require extremely small segments, which would impose substantial overhead.

6 Conclusion and Future Work

In this paper, we propose an open-source WebRTC-based selective forwarding unit (SFU) for many-to-many volumetric conferencing. Additionally, we present a reference client implementation based on the VR2Gather platform, with support for VR2Gather's existing quality adaptation. In networks without latency, our approach achieves similar or better performance compared to the existing TCP-based protocols in VR2Gather. With 50 ms of induced latency, the proposed SFU reduces latency by 21% compared to TCP Reflector while matching the other protocols. Using publicly available captured videos and movement traces, we also show that our WebRTC SFU combined with VR2Gather's quality adaptation yields faster visible quality updates: 13% and 36% improvements at 5 ms and 10 ms induced latency, respectively. Moreover, small 5 ms latency variations increase the time it takes for a quality switch to become visible for Direct TCP latency by 35%, whereas WebRTC maintains stable performance.

In future work, we will extend the benchmark to a larger number of users to better assess the scalability of the system. The current quality adaptation algorithm of VR2Gather assumes fixed bandwidth, which limits applicability to real-world conditions. Since the platform already supports bandwidth estimation, integrating server-side quality decisions would improve adaptability to varying network conditions.

References

[1] Anonymous. 2024. Anonymous. https://removed.double.blind/". Accessed: 2024-08-28. (2024).

[2] C. Cortés, I. Viola, J. Gutiérrez, J. Jansen, S. Subramanyam, E. Alexiou, P. Pérez, N. García, and P. César. 2024. Delay threshold for social interaction in volumetric extended reality communication. *ACM Transactions on Multimedia Computing, Communications and Applications*, 20, 7, 1–22.

[3] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou. 2017. 8i Voxelized Full Bodies - A Voxelized Point Cloud Dataset. *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006*, 7, 8.

[4] M. De Fré, J. van der Hooft, C.-Y. Chang, K. De Schepper, P. R. Alface, D. De Vleeschauwer, T. Wauters, P. Steenkiste, and F. De Turck. 2025. Low-latency volumetric video conferencing in congested networks through l4s. In *Proceedings of the 16th ACM Multimedia Systems Conference*, 113–123.

[5] M. De Fré, J. van der Hooft, T. Wauters, and F. De Turck. 2024. Scalable MDC-based volumetric video delivery for real-time one-to-many WebRTC conferencing. In *Proceedings of the 15th ACM Multimedia Systems Conference*, 121–131.

[6] S. Dutton. 2013. Webrtc in the real world: stun, turn and signaling. *Google, Nov*, 1–22.

[7] Google. 2024. Draco. https://google.github.io/draco/. Accessed: 2024-09-11. (2024).

[8] Google. 2024. Real-Time Communication for the Web. https://webrtc.org/. Accessed: 2024-08-30. (2024).

[9] D. Graziosi, O. Nakagami, S. Kuma, A. Zaghetto, T. Suzuki, and A. Tabatabai. 2020. An Overview of Ongoing Point Cloud Compression Standardization Activities: Video-Based (V-PCC) and Geometry-Based (G-PCC). *APSIPA Transactions on Signal and Information Processing*, 9.

[10] B. Grozev, G. Politis, E. Ivov, T. Noel, and V. Singh. 2017. Experimental evaluation of simulcast for webrtc. *IEEE Communications Standards Magazine*, 1, 2, 52–59.

[11] S. N. B. Gunkel, R. Hindriks, K. M. E. Assal, H. M. Stokking, S. Dijkstra-Soudarissanane, F. Haar, and O. Niamut. 2021. VRComm: An End-to-End Web System for Real-Time Photorealistic Social VR Communication. In *ACM Multimedia Systems Conference*.

[12] J. Jansen, T. Röggla, S. Rossi, I. Viola, and P. Cesar. 2024. Open-Sourcing VR2Gather: A Collaborative Social VR System for Adaptive Multi-Party Real Time Communication. In *ACM International Conference on Multimedia*.

[13] J. Jansen, S. Subramanyam, R. Bouqueau, G. Cernigliaro, M. Cabré, F. Pérez, and P. Cesar. 2020. A Pipeline for Multiparty Volumetric Video Conferencing: Transmission of Point Clouds Over Low Latency DASH. In *ACM Multimedia Systems Conference*.

[14] J. Lawrence et al. 2021. Project Starline: A High-Fidelity Telepresence System. *ACM Transactions on Graphics*, 40, 6.

[15] L.-H. Lee, T. Braud, P. Y. Zhou, L. Wang, D. Xu, Z. Lin, A. Kumar, C. Bermejo, P. Hui, et al. 2024. All one needs to know about metaverse: a complete survey on technological singularity, virtual ecosystem, and research agenda. *Foundations and Trends® in Human-Computer Interaction*, 18, 2–3, 100–337.

[16] R. Mekuria, K. Blom, and P. Cesar. 2016. Design, Implementation, and Evaluation of a Point Cloud Codec for Tele-Immersive Video. *IEEE Transactions on Circuits and Systems for Video Technology*, 27, 4.

[17] MIT. 2024. Pion WebRTC. https://github.com/pion/webrtc". Accessed: 2024-09-11. (2024).

[18] J. Park, P. A. Chou, and J.-N. Hwang. 2019. Rate-utility optimized streaming of volumetric media for augmented reality. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9, 1, 149–162.

[19] [SW], Anonymous 2023. URL: Anonymous.

[20] [SW], Anonymous 2023. URL: Anonymous.

[21] B. Sredojev, D. Samardzija, and D. Posarac. 2015. Webrtc technology overview and signaling solution design and implementation. In *2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO)*. IEEE, 1006–1009.

[22] S. Subramanyam, I. Viola, A. Hanjalic, and P. Cesar. 2020. User Centered Adaptive Streaming of Dynamic Point Clouds with Low Complexity Tiling. In *ACM International Conference on Multimedia*.

[23] S. Subramanyam, I. Viola, J. Jansen, E. Alexiou, A. Hanjalic, and P. Cesar. 2022. Evaluating the Impact of Tiled User-Adaptive Real-Time Point Cloud Streaming on VR Remote Communication. In *ACM International Conference on Multimedia*.

[24] J. van der Hooft, H. Amirpour, M. Torres Vega, Y. Sanchez, R. Schatz, T. Schierl, and C. Timmerer. 2023. A Tutorial on Immersive Video Delivery: From Omnidirectional Video to Holography. *IEEE Communications Surveys & Tutorials*.

[25] J. van der Hooft, T. Wauters, F. De Turck, C. Timmerer, and H. Hellwagner. 2019. Towards 6dof http adaptive streaming through point cloud compression. In *Proceedings of the 27th ACM International Conference on Multimedia*, 2405–2413.

[26] I. Viola, J. Jansen, S. Subramanyam, I. Reimat, and P. Cesar. 2023. VR2Gather: A Collaborative Social VR System for Adaptive Multi-Party Real-Time Communication. *IEEE MultiMedia*.

[27] I. Viola and P. Cesar. 2023. Volumetric video streaming: current approaches and implementations. *Immersive Video Technologies*, 425–443.

[28] P. K. Yadav and W. T. Ooi. 2020. Tile rate allocation for 360-degree tiled adaptive video streaming. In *Proceedings of the 28th ACM International Conference on Multimedia*, 3724–3733.