Distributed WebRTC-Based Forwarding for Scalable Volumetric Video Streaming

Anonymous Author(s)

Abstract

As immersive media becomes increasingly accessible, virtual counterparts to real-world experiences such as concerts and conferences have emerged, enabled by volumetric streaming pipelines for virtual reality (VR). User representation is central to these systems, with point clouds widely adopted for realistic avatars due to their balance between quality and performance. However, most existing systems struggle to scale to larger user counts. While recent work has proposed more scalable architectures, these typically focus on computational optimizations and fail to scale to high user counts due to network bottlenecks. To address this gap, we present an open-source, modular, distributed WebRTC-based volumetric streaming pipeline that employs multiple selective forwarding units (SFUs) to improve latency, throughput, and visual quality compared to a centralized SFU. Results show that, with 64 users, the distributed architecture receives 147% more points while maintaining comparable transport latency, and reduces peak latency at lower user counts. Furthermore, leveraging quality adaptation enables stable transport latency across scales, achieving approximately 25 ms .

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. Distributed WebRTC-Based Forwarding for Scalable Volumetric Video Streaming. 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

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

ACM ISBN TODO

https://doi.org/TODO

1 Introduction

In recent years, immersive media has expanded across both entertainment and professional domains, with improvements in network performance and content delivery [21]. The availability of lightweight virtual reality (VR) head-mounted displays (HMDs) has further accelerated adoption, positioning immersive media as a medium for education, remote collaboration, and virtual conferencing [32, 20]. In immersive systems, user representation varies by application requirements: collaborative use cases often employ low-fidelity synthetic avatars to support real-time interaction [37], while virtual conferencing prioritizes visual realism [35]. In general, such scenarios remain constrained in scalability, typically supporting only a limited number of participants. Recent work has addressed these limitations through more efficient network architectures and rendering pipelines [11]. These advances facilitate complex large-scale virtual environments, though they require sophisticated mechanisms to balance scalability, latency, and quality across clients [15]. The choice of transport protocol significantly impacts the performance and scalability of immersive systems. Transmission control protocol (TCP)-based solutions such as low-latency DASH (LL-DASH) offer reliable delivery and adaptive quality but suffer from high latency and limited interactivity [33]. In contrast, user datagram protocol (UDP)-based protocols including WebRTC focus on achieving the lowest latency, at the cost of lacking reliable delivery. Scalable architectures built on these protocols further increase the number of concurrent users [2, 27], though achieving true global-scale deployment remains challenging. To address these shortcomings, this paper proposes a novel many-to-many volumetric video conferencing pipeline using distributed WebRTC selective forwarding units (SFUs) to increase scalability and overall performance, containing the following contributions:

- An open-source [28] WebRTC-based many-to-many volumetric streaming pipeline leveraging multiple SFUs for bandwidth-efficient distributed streaming;
- A large-scale evaluation analyzing scalability across varying numbers of interconnected SFUs and clients.

2 Related Work

This section reviews related work on scalable volumetric video delivery, focusing on design choices affecting quality and real-time transmission, and scalable architectures.

Volumetric Content Delivery. Real-time immersive systems often adopt point cloud representations due to their simplicity and efficiency [10]. While point clouds are well suited for virtual conferencing, their high bitrates hinder real-time transmission [16]. Certain codecs achieve video-like bitrates but incur high latency [22, 29]. In contrast, real-time codecs such as Draco [14] trade increased bandwidth for low latency. Moreover, bandwidth can be further reduced with quality adaptation techniques such as tilling [30] and scalable encoding [12], enabling dynamic adjustment based on user field of view (FoV) and network conditions, thereby reducing bitrates while preserving visual quality [18]. For these reasons, we will be using point clouds encoded with Draco to ensure scalable, real-time performance.

Low-Latency Video Streaming. Several immersive systems have adapted traditional transport solutions to support volumetric content [5]. For scenarios that require high reliability, TCP-based protocols, such as LL-DASH, are often chosen. However, even this low latency variant still experiences high latencies, limiting interactivity [6]. In contrast, UDP-based solutions, such as WebRTC, prioritize low latency over strict reliability, although such features can be implemented at the application level [25]. Although UDP-based methods do not inherently support bandwidth estimation, algorithms such as Google congestion control (GCC) [8] are frequently employed to provide this functionality. Based on the estimated bandwidth, and potentially the user's FoV, adaptive algorithms then determine the most suitable quality for each user [34]. More advanced approaches further refine this process by interleaving frame types or combining multiple quality levels to optimize visual quality [7, 36]. For these reasons, we will utilize WebRTC together with a bandwidth estimator to enable low-latency adaptive streaming.

Scalable Streaming Architectures. Scenarios involving a larger number of clients often employ edge servers that are located closer to the clients to increase scalability [23, 26]. Some systems distinguish between sender and receiver connections. In these architectures, a client uploads its content to a single server but receives content from multiple servers [19]. This design helps address the imbalance between limited user upload bandwidth and comparatively larger download capacity. However, if many clients request the same content from a single server, that server becomes a bottleneck [9]. Furthermore, when multiple nearby clients request identical content from a distant server, the same data must traverse long network paths, increasing the contention on shared links and queuing delays [38]. Other distributed systems, restrict each user to a single bidirectional connection, with the connected server being responsible to forward the content to the other servers. Such systems introduce additional latency due to each server having to process each packet [13]. However, as clients are spread across all servers, the overall computational load is also reduced. When multiple clients connected to the same server request identical content, forwarding-based architectures are advantageous, as content can be relayed over high-capacity links, with only the final hop to clients relying on consumer connections [1]. To increase scalability and to reduce bandwidth waste over shared network links, a forwarding approach will be used.

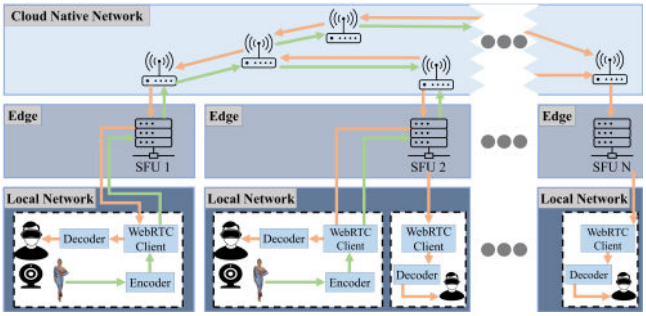

Figure 1: Proposed distributed SFU architecture for many-to-many volumetric streaming, supporting content forwarding between multiple SFUs. Each client can either serve as a sender, receiver or both.

3 System Architecture

This section introduces the proposed scalable, distributed SFU-based WebRTC pipeline (Figure 1), outlining the streaming workflow, session management, low-latency WebRTC mechanisms, the session description protocol (SDP) exchange in a distributed setting, and the core SFU design choices.

Volumetric Video Pipeline. First content is captured using a depth camera to acquire color and depth frames and convert them into a point cloud. The resulting point cloud is then partitioned into multiple descriptions to enable quality adaptation and parallel encoding. Finally, the encoded point cloud is sent to the WebRTC SFU. At the receiver, the descriptions are delivered by the SFU, decoded in parallel, merged into a single point cloud, and displayed on an HMD.

Session Manager. The session manager is responsible for session coordination and provisioning of SFUs. Before transmitting content, clients register their tracks and parameters, ensuring a consistency across clients and SFUs. The session manager then instructs the target SFU to allocate resources and, once ready, returns the connection details to the client.

Real-Time Scalable Delivery with WebRTC. Multipoint control unit (MCU)-based systems aggregate streams into a single composite output, offering improved synchronization and reduced bandwidth at the cost of increased latency. In contrast, SFU-based systems forward packets directly to subscribed clients enabling lower latency and better scalability.

Although WebRTC does not provide inherent reliable delivery, negative acknowledgment (NACK)-based retransmissions are commonly used to reduce undecodable frames. Unlike TCP reliability, this approach allows applications to trade reliability for latency by lowering the number of retransmit requests. To prevent packet loss, WebRTC-based systems employ congestion control algorithms, such as GCC, enabling low-latency under varying network conditions. These controllers estimate available bandwidth and delay, allowing applications to adapt content before the network degrades.

SDP Message Exchange. In WebRTC, connections are established through the exchange of SDP messages that negotiate supported codecs, transport parameters, and network candidates [24]. Negotiation uses a state machine that prevents renegotiation until the current exchange completes. While this process is straightforward in single SFU architectures, extending it to multiple SFUs introduces additional coordination challenges. When a new client connects to an SFU, all other SFUs must be notified via SDP renegotiation that new WebRTC tracks are available. Concurrent renegotiation attempts can drive the negotiation state machine into invalid states. To avoid such conflicts, a deterministic ordering, such as using an ID, among SFUs is required to decide which node initiates the SDP exchange and which responds.

Distributed SFU Design. By default, clients connected to one SFU cannot receive content from clients connected to another. To enable cross-SFU communication, all SFUs within a session establish interconnections and forward relevant audio and video packets from clients to interested SFUs. This process is transparent to clients, which maintain a single SFU connection, preserving compatibility with existing WebRTC implementations. The SFU is content-agnostic and supports transmission of volumetric media encoded with arbitrary codecs. While the SFU remains content-independent, a quality adaptation module tailors the transmitted content based on bandwidth estimates, ranging from a best-effort track selection to multiple description coding (MDC)-based approaches that balance quality across visible clients.

4 Implementation and Evaluation

This section describes the implementation of the distributed pipeline, the evaluation scenarios and quality adaptation methods, the considered metrics, and the results.

4.1 Implementation Details

The pipeline is composed of several modules. At the center is the graphical interface, implemented in Unity [31], which renders volumetric content and displays it in VR. The application interfaces with a depth camera module that generates point clouds, which are subsequently encoded using an MDC-based encoder and compressed with Draco. The capturing and encoding components are implemented in C++, while the WebRTC peer, SFU, and session manager are implemented in Golang, with Unity integration provided via a C++ library. The pipeline supports both headful and headless modes. In headful mode, live data is captured and rendered in VR, whereas headless mode disables capture and rendering and may use pre-encoded content to focus on networking and quality adaptation performance. Since this evaluation focuses on comparing centralized and distributed WebRTC SFU network architectures, most experiments use headless mode with pre-encoded data, with a full end-to-end evaluation being presented in Section 4.7. GCC was selected as the congestion control algorithm, with the minimum bitrate configured to 15 Mbit/s. Based on the bandwidth estimates provided by GCC, can be adapted with different methods. The specific adaptation strategy depends on the evaluation scenarios described in Section 4.3.

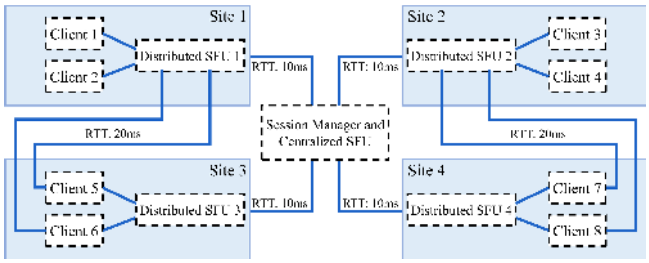

Figure 2: Bare-metal setup used for experimentation. Each SFU node can either serve as a normal SFU or as a simple switch node and each client node can host one or more WebRTC client applications.

4.2 Experimental Setup

The experiments are carried out on [anon] [3], a testbed of interconnected bare-metal nodes. Each node runs Ubuntu 22.04 and has the following specifications: *CPU: Intel i5-9500 (3.0GHz), RAM: 64GB DDR4 (3200 MHz), 5 NICs:* 1 Gbit/s. Figure 2 shows the physical network topology. The topology comprises four sites emulating distinct locations, interconnected via a central node. Additional direct links connect Site 3 to Site 1 and Site 4 to Site 2, providing alternative routing paths that bypass the shared node. Inter-site latency is emulated using traffic control (tc). Client nodes host one or more headless client instances, while SFU nodes either run SFU instances or act as routers, depending on the experiment. As the evaluation focuses on network scalability, clients omit encoding, decoding, and rendering. Each client uses a 1,800-frame sequence, with approximately 100,000 points per frame, captured with an Intel RealSense D455 camera [17] at a resolution of 848×484 and 30 FPS [4]. Additionally, the point clouds are encoded with the Draco codec using 11-bit quantization and maximum encoding speed.

4.3 Evaluation Scenarios

Two virtual conferencing scenarios are considered. For each scenario, multiple configurations are evaluated, comparing a centralized SFU baseline with distributed architectures using two SFUs (Sites 1 and 2) and four SFUs (one per site). Each configuration is tested with 8 to 64 users in increments of 8. Experiments are repeated 20 times and run for 120 s, with the first 60 s used for session setup and GCC convergence.

Scenario 1, high-scale, base-quality-only streaming: This scenario evaluates scalability when each client transmits only a base-quality point cloud (15% of points, approximately 12 Mbit/s). All clients receive streams from every other client, generating a large number of flows that compete for network resources. Quality adaptation relies on GCC bandwidth estimates, which are used to randomly select a subset of streams to be forwarded to each client. Because every client stream has the same bitrate, the incremental bandwidth required to support an additional user remains constant, ensuring a more gradual increase in outgoing throughput.

Scenario 2, small-scale, high-quality adaptive streaming: This scenario models a debate- or concert-like setting in which four clients (one per site) act as senders, each transmitting three MDC-encoded point cloud representations (15%, 25%, and 60% of the original point cloud at 12 Mbit/s, 18 Mbit/s, and 42 Mbit/s, respectively), which can be combined at the receivers. Unlike Scenario 1, many clients will receive similar data due to the bitrate assignment being independent of the number of clients, as in all configurations exactly four clients transmit three MDC-encoded representations. However, the bitrate gaps between quality levels are non-uniform, which challenges GCC to keep latency low.

4.4 Evaluation Metrics

For each evaluation scenario mentioned in the previous section, the following metrics are observed and discussed:

First, achieved throughput and packet loss.

Second, in Scenario 1, quality is determined by the number of received client representations, while in Scenario 2, quality is assessed by the number of points in MDC-encoded clouds.

Third, a full end-to-end breakdown of the system is provided, with a focus on the transport components.

Fourth, transport latency is further analyzed by emphasizing worst-case behavior during network congestion.

Fifth, CPU and memory usage are examined.

4.5 Throughput and Packet Loss

Figure 3 shows the throughput consumed by clients across the network, excluding traffic required to send data from a client to the SFU or to forward content between SFUs. Distributed deployments achieve significantly higher throughput due to reduced link sharing among clients. In contrast,

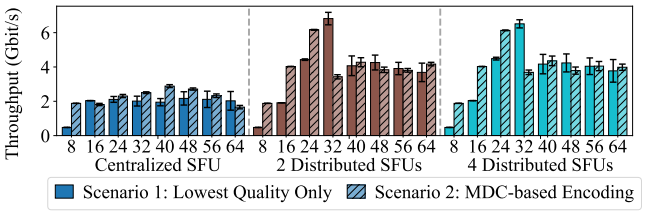

Figure 3: Total throughput of client data, with 95% confidence intervals, for user counts of 8 to 64.

the centralized SFU reaches its maximum throughput with only eight clients in the MDC-based configuration, after which additional clients must contend for the same capacity.

The distributed setups exhibit comparable throughput. Because the setup with four SFUs forwards data to fewer clients, it can be concluded that throughput is not constrained by the processing capacity required to serve a given number of clients. Instead, both distributed setups achieve similar throughput primarily due to the experimental topology. In the two-SFU configuration, clients in Site 3 and Site 4 are directly connected to the SFUs in Site 1 and Site 2, enabling higher throughput through unshared links (Figure 2). Removing these links would reduce the available capacity by forcing traffic to share the same links. The four-SFU setup avoids this bottleneck by colocating each client with a local SFU, suggesting that deploying SFUs closer to clients substantially increases throughput. Combining the throughput in Figure 3 with the used topology suggests that approximately 50% of a link's bandwidth can be used by WebRTC before delay becomes excessive. Both distributed setups also exhibit a throughput spike at 24 users in Scenario 2 and at 32 users in Scenario 1. This behavior suggests that an ideal user count exists in which available bandwidth is utilized efficiently, without excessive interference among flows.

Because GCC reacts preemptively to increases in inter-packet delay, packet loss remains low, averaging between 0% and 4% across all setups. However, since GCC does not account for the simultaneous bandwidth increases of multiple streams, short periods of substantial packet loss (>20%) may occur. These congestion events are brief, as the packet loss causes GCC to reduce its estimate. Moreover, NACK-based retransmissions, limit the number of undecodable frames.

4.6 Quality and Number of Received Clients

Network throughput directly impacts per-client quality. In the first scenario, each client can receive point clouds from all other participants; however, since only a single quality representation is transmitted per sender, a client either receives a complete point cloud or none at all. Quality is therefore measured as the ratio of successfully received point clouds to the maximum possible number (i.e., the total number of clients minus one), as reported in Table 1. For 8 and 16 users, both configurations deliver all point clouds to every client. As

Table 1: Scenario 1 quality, calculated based on the number of received point clouds compared to the maximum receivable amount. Both distributed SFU configuration achieve significantly better performance compared to the centralized SFU, with high standard deviations caused by competing flows.

#Clients	8		16		24		32		40		48		56		64	
%Of Maximum Quality	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
Centralized SFU	100.0	0.0	100.0	0.0	43.3	29.6	23.1	21.9	14.3	17.0	11.0	15.0	33.6	17.9	18.9	9.7
2 Distributed SFUs	100.0	0.0	100.0	0.0	90.8	18.8	78.8	31.25	30.8	31.3	21.7	21.5	57.3	19.3	47.7	18.0
4 Distributed SFUs	100.0	0.0	100.0	0.0	91.7	7.9	75.0	24.7	32.3	33.8	21.7	22.3	60.5	17.1	46.9	23.4

Table 2: Scenario 2 quality, calculated based on the number of points in the merged point clouds created from MDC representations compared to the points in the original clouds. Both distributed configurations achieve significantly better performance compared to the centralized SFU, with high standard deviations caused by competing flows.

#Clients	8		16		24		32		40		48		56		64	
%Of Maximum Quality	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
Centralized SFU	100.0	0.0	43.8	17.3	38.7	18.5	36.9	7.8	34.0	8.5	26.9	8.3	18.8	10.0	12.1	6.2
2 Distributed SFUs	100.0	0.0	100.0	0.0	99.8	2.0	41.3	18.8	38.9	19.4	35.9	16.7	32.1	10.8	30.5	11.5
4 Distributed SFUs	100.0	0.0	100.0	0.0	99.2	3.6	42.9	15.8	39.8	20.8	34.2	14.8	33.9	9.6	30.0	15.0

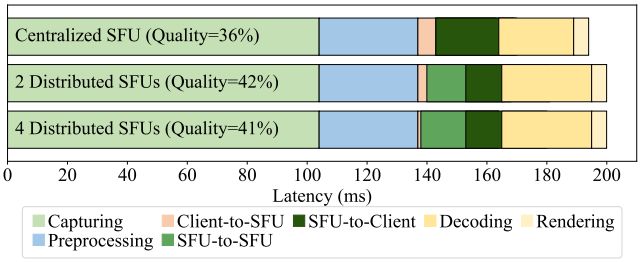

Figure 4: Average end-to-end latency per sender in an MDC-based scenario with 32 clients and four senders. The WebRTC latency is broken down in 3 categories.

the number of users increases, the centralized setup exhibits a decline in received point clouds, whereas the distributed configurations sustain up to 3.5 times more point clouds. Noticeable degradation in the distributed setups occurs only at 40 or more users, where clients still receive approximately twice as many point clouds as in the centralized case.

Scenario 2 employs MDC-based encoding which enables clients to receive multiple quality representations of a point cloud. Since each representation contributes distinct points, quality is defined as the ratio of points in the merged point cloud to those in the original. The final quality score is computed as the average across all received point clouds, with missing point clouds contributing 0%. As shown in Table 2, both distributed setups achieve substantially higher quality than the centralized SFU. The relatively large standard deviation is primarily due to the limited set of seven distinct MDC combinations, which differ by at least 10% of the points.

4.7 End-to-End Latency Breakdown

Figure 4 shows the average end-to-end latency and quality for one of the four sender nodes in Scenario 2 with 32 participants. The centralized setup achieves slightly lower latency due to reduced decoding overhead resulting from the lower throughput discussed in Section 4.5, which limits delivery to lower-quality representations. Focusing on transport-related components (Client–SFU, SFU–SFU, and SFU–Client), all setups differ by only a few milliseconds. These components are measured as the time between sending the first packet and receiving the last packet. A small difference is observed between the 2-SFU and 4-SFU setups: the higher client-to-SFU latency in the 2-SFU case results from routing, as clients in Sites 3 and 4 must transmit data to SFUs located in other sites. Consequently, traffic may traverse multiple hops, each adding delay (Figure 2). This effect does not occur in the centralized setup, where all traffic follows a path through the central node. Nevertheless, when sender and receiver are co-located, both distributed setups achieve lower latency than the centralized setup by avoiding inter-site links altogether.

However, the 4-SFU configuration forwards more data (210 Mbit/s when one client's stream is forwarded to three SFUs, compared to 140 Mbit/s when two clients are forwarded to a single SFU), slightly increasing forwarding-induced network latency. This trade-off underscores the importance of both SFU count and placement, and suggests that dynamic orchestration could adapt deployments based on current network conditions and client locations.

The camera introduces the most latency because of the delay between image capture and processing. Because of MDC-based encoding, preprocessing latency remains constant across setups and is independent of the number of clients. In contrast, decoding latency increases with the number of descriptions received from the other senders.

4.8 Peak Transport Latency

Large latency spikes can severely diminish immersion, and even brief, inconsistent delays can disrupt interaction. Figure 5 presents the median network latency alongside the

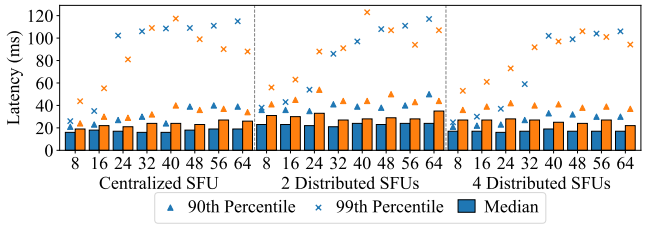

Figure 5: Median latency is stable for user counts of 8 to 64. Peak latencies, represented by the 90th and 99th percentiles, increase more but still remain stable.

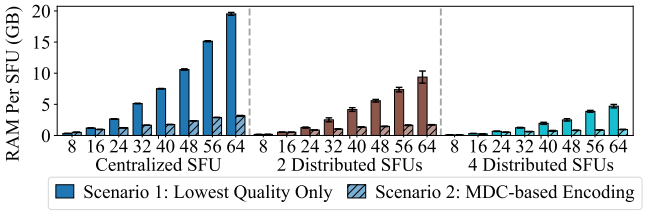

Figure 6: RAM usage per SFU, with 95% confidence intervals, for user counts of 8 to 64.

90th- and 99th-percentile values, which together indicate both the frequency and severity of latency spikes.

As shown in Figure 5, median latency remains stable, increasing only slightly with the number of users. The centralized setup exhibits the lowest median latency when compared to the 2-SFU setup. However, with four SFUs, latency becomes comparable to the centralized case due to improved routing when senders and receivers are co-located. Consistent with the results in Section 4.5, GCC contributes to stable latency. As discussed in Section 4.6, this stability is achieved by aggressively reducing bandwidth estimates upon congestion, which directly lowers quality. Nevertheless, despite GCC, large spikes may still occur when multiple flows simultaneously increase their rates. The 99th-percentile results show that, due to more efficient bandwidth utilization, the distributed setups exhibit a smoother increase in worst-case latency as the number of users grows. In the centralized setup, latency spikes emerge at 24 users, whereas the distributed setups experience only similar values at 40 users. Consistent with the median, the 90th-percentile latency remains stable across all setups. Overall, these results indicate that the worst-case latency remains acceptable for user interaction.

4.9 System Resources

Figure 6 shows that, memory usage in both scenarios is substantially higher for the centralized SFU than for both distributed setups. While each distributed SFU individually consumes less memory, the aggregate usage across all distributed SFUs is comparable to, or slightly lower than, that of a centralized SFU. In Scenario 1, the high usage is attributed to the NACK-based retransmissions. With NACK enabled, every packet sent to at least one client must be temporarily

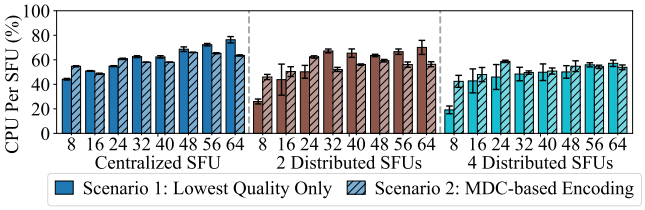

Figure 7: Average CPU usage across all cores, with 95% confidence intervals, for user counts of 8 to 64.

buffered. In the centralized setup, serving all clients and forwarding many distinct tracks increases the number of buffers, causing memory usage to grow rapidly with user count. In contrast, when clients receive identical data, as in Scenario 2, buffers can be shared, keeping memory usage low. Furthermore, reducing buffer sizes is impractical, as the high bitrate of point clouds would cause buffers to overflow rapidly. Figure 7 shows that CPU usage only modestly increases as the number of users grows. An exception occurs in Scenario 2: with eight users, CPU usage is significantly higher than in Scenario 1, reflecting the higher initial bitrate of a single MDC-encoded object (70 Mbit/s vs. 12 Mbit/s). As discussed in Section 4.5, GCC subsequently stabilizes throughput and latency, causing CPU usage to converge once the user count exceeds 32. The CPU spikes observed at 24 and 32 users align with the throughput spikes in Figure 3.

5 Conclusion and Future Work

This paper presented a scalable many-to-many volumetric video streaming architecture that leverages distributed selective forwarding units (SFUs) for WebRTC to improve performance, increase supported user counts, and reduce resource utilization. We evaluated centralized and distributed setups with two and four SFUs under user counts ranging from 8 to 64 across two scenarios: high scale, low-quality point cloud streaming, and selective multiple description coding (MDC)-based multi-quality transmission with limited senders. Results show that distributed setups utilize bandwidth more efficiently, doubling throughput by reducing shared links. This enables greater client coverage in the first scenario and substantially higher quality in the second, with up to 147% more received points with 64 users. While Google congestion control (GCC) maintains low packet loss and stable latency across all setups, worst-case latency differs markedly: the centralized setup exceeds 100 ms at 32 users, whereas distributed setups only reach this threshold beyond 48 users.

Future work will extend the architecture to a fully hierarchical topology, enabling multi-level forwarding across wider regions, supporting user counts beyond 64. Dynamic SFU scaling will also be explored with seamless client handover, and adaptive client-to-SFU assignment based on metrics such as latency and the current field of view (FoV) of all users.

References

[1] A. O. Al-Abbasi, V. Aggarwal, and M.-R. Ra. 2019. Multi-tier caching analysis in cdn-based over-the-top video streaming systems. *IEEE/ACM Transactions on Networking*, 27, 2, 835–847.

[2] E. André, N. Le Breton, A. Lemesle, L. Roux, and A. Gouaillard. 2018. Comparative study of webrtc open source sfus for video conferencing. In *2018 Principles, Systems and Applications of IP Telecommunications (IPTComm)*. IEEE, 1–8.

[3] Anonymous. 2025. Anonymous. https://removed.double.blind/". Accessed: 2025-11-14. (2025).

[4] Anonymous. 2025. Anonymous. https://removed.double.blind/". Accessed: 2025-11-14. (2025).

[5] A. Bentaleb, M. Lim, S. Hammoudi, S. Harous, and R. Zimmermann. 2025. Solutions, challenges, and opportunities in volumetric video streaming: an architectural perspective. *ACM Transactions on Multimedia Computing, Communications and Applications*, 21, 7, 1–35.

[6] A. Bentaleb, Z. Zhan, F. Tashtarian, M. Lim, S. Harous, C. Timmerer, H. Hellwagner, and R. Zimmermann. 2022. Low latency live streaming implementation in dash and hls. In *Proceedings of the 30th ACM International Conference on Multimedia*, 7343–7346.

[7] B. Bross, Y.-K. Wang, Y. Ye, S. Liu, J. Chen, G. J. Sullivan, and J.-R. Ohm. 2021. Overview of the versatile video coding (vvc) standard and its applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 31, 10, 3736–3764.

[8] G. Carlucci, L. De Cicco, S. Holmer, and S. Mascolo. 2016. Analysis and design of the google congestion control for web real-time communication (WebRTC). In *Proceedings of the 7th International Conference on Multimedia Systems*, 1–12.

[9] I. Chicano-Capelo, F. Gortázar, and M. Gallego. 2025. Quality of experience under huge load for webrtc applications: a case study of three media servers. *IEEE Access*.

[10] J. Choi, J.-B. Jeong, S. Lee, and E.-S. Ryu. 2022. Overview of the volumetric video capturing system for immersive media. In *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 574–577.

[11] M. Dasari, E. Lu, M. W. Farb, N. Pereira, I. Liang, and A. Rowe. 2023. Scaling vr video conferencing. In *2023 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*. IEEE, 648–657.

[12] 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.

[13] R. Deshmukh, N. Nand, A. Pawar, D. Wagh, and A. Kudale. 2023. Video conferencing using webrtc. In *2023 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS)*. IEEE, 857–864.

[14] Google. 2023. Draco. (2023). https://google.github.io/draco/.

[15] D. Dziwis and H. von Coler. 2023. The entanglement: volumetric music performances in a virtual metaverse environment. *Journal of Network Music and Arts*, 5, 1, 3.

[16] M. Hosseini and C. Timmerer. 2018. Dynamic adaptive point cloud streaming. In *Proceedings of the 23rd Packet Video Workshop*, 25–30.

[17] [SW], Intel Realsense 2025. URL: https://www.intelrealsense.com/.

[18] Y. Jin, K. Hu, J. Liu, F. Wang, and X. Liu. 2023. From capture to display: a survey on volumetric video. *arXiv preprint arXiv:2309.05658*.

[19] R. A. Kırmızıoğlu, A. M. Tekalp, and B. Görkemli. 2025. Distributed virtual selective-forwarding units and sdn-assisted edge computing for optimization of multi-party webrtc videoconferencing. *Signal Processing: Image Communication*, 130, 117173.

[20] S. F. Langa, M. Montagud, G. Cernigliaro, and D. R. Rivera. 2022. Multiparty holomeetings: toward a new era of low-cost volumetric holographic meetings in virtual reality. *Ieee Access*, 10, 81856–81876.

[21] 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.

[22] G. Li, W. Gao, and W. Gao. 2024. Mpeg geometry-based point cloud compression (g-pcc) standard. In *Point Cloud Compression: Technologies and Standardization*. Springer, 135–165.

[23] S. Maheshwari, D. Raychaudhuri, I. Seskar, and F. Bronzino. 2018. Scalability and performance evaluation of edge cloud systems for latency constrained applications. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 286–299.

[24] S. A. Mahmood and M. Kherallah. 2023. Evaluation of the existing web real-time signaling mechanism for peer-to-peer communication: survey. In *International Conference on Advanced Engineering, Technology and Applications*. Springer, 298–310.

[25] P. Malinovskiy. 2025. Solving the problem of poor internet connectivity in dhaka: innovative solutions using advanced webrtc and adaptive streaming technologies. *arXiv preprint arXiv:2506.17343*.

[26] M. Medvetskyi, M. Beshley, I. Scherm, O. Urikova, H. Beshley, and T. Dierkes. 2024. Sdn-based 6g video content delivery technique via multi-path routing, multi-server architecture and multi-connectivity. In *2024 IEEE 17th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*. IEEE, 169–173.

[27] P. Nuño, F. G. Bulnes, J. C. Granda, F. J. Suárez, and D. F. García. 2018. A scalable webrtc platform based on open technologies. In *2018 International Conference on Computer, Information and Telecommunication Systems (CITS)*. IEEE, 1–5.

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

[29] M. Rudolph et al. 2024. Transcoding v-pcc point cloud streams in real-time. *ACM Transactions on Multimedia Computing, Communications and Applications*.

[30] S. Subramanyam, I. Viola, A. Hanjalic, and P. Cesar. 2020. User centered adaptive streaming of dynamic point clouds with low complexity tiling. In *Proceedings of the 28th ACM international conference on multimedia*, 3669–3677.

[31] [SW], Unity 2023. URL: https://unity.com.

[32] N. Van der Meer, V. van der Werf, W.-P. Brinkman, and M. Specht. 2023. Virtual reality and collaborative learning: a systematic literature review. *Frontiers in Virtual Reality*, 4, 1159905.

[33] J. van der Hooft, H. Amirpour, M. T. 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, 2, 1336–1375.

[34] 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.

[35] 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*, 30, 2, 48–59.

[36] Y. Wang, A. R. Reibman, and S. Lin. 2005. Multiple description coding for video delivery. *Proceedings of the IEEE*, 93, 1, 57–70.

[37] S. M. Yamijala, N. Nimbrain, and R. Bansal. 2025. Enhancing remote workspaces: the role of virtual reality in shaping the virtual work environment. *Optimizing virtual reality and metaverse for remote work and virtual team collaboration*, 171–190.

[38] D. Yuan et al. 2025. Understanding operational cdn live streaming: a measurement study on performance, costs and enhancements. *IEEE Transactions on Circuits and Systems for Video Technology*.