Towards Scalable, Low-Latency Volumetric Streaming: A Hybrid Predictive Synchronization Framework

Towards Scalable, Low-Latency Volumetric Streaming: A Hybrid Predictive Synchronization Framework

Robin Singh Sidhu*, Xiang Su†‡✉, Xiaoli Liu§†, Hao Wang¶, and Faouzi Alaya Cheikh*
*Department of Computer Science, Norwegian University of Science and Technology, Gjøvik, Norway
†Department of Computer Science, University of Helsinki, Helsinki, Finland
‡Department of Agricultural Sciences, University of Helsinki, Helsinki, Finland
§School of Technology and Innovations, University of Vaasa, Vaasa, Finland
¶School of Cyber Engineering, Xidian University, Xi'an, China
robin.s.sidhu@ntnu.no, ✉xiang.su@helsinki.fi, xiaoli.liu@helsinki.fi, haow@ieee.org, faouzi.cheikh@ntnu.no

***Abstract*—Volumetric video is emerging as a cornerstone for multi-user Extended Reality (XR) and metaverse applications. However, achieving synchronous playback across distributed clients remains challenging under heterogeneous network conditions (such as varying latency, jitter, packet loss, and asymmetric bandwidth). Existing synchronization approaches, such as state synchronization and fixed-frame buffering, struggle with scalability and often trade off latency for playback continuity. We present a predictive synchronization framework that combines lightweight time-series latency forecasting with adaptive buffering control. Our framework incorporates a hybrid predictor that adaptively selects the most suitable model (including EWMA, ARIMA, LSTM) for each client, escalating to heavier predictors only when residual errors exceed thresholds. This design balances accuracy with computational overhead, enabling lower latencies at larger scales. Our experimental results show that predictive synchronization reduces average inter-client skew by up to 76% compared to baselines, while also decreasing the average buffer size by 40% in a controlled multi-client testbed. The average per-frame latency remains below 20 ms and the 95th-percentile latency below 70 ms for up to 100 concurrent clients. These results demonstrate that prediction-aware, hybrid synchronization substantially improves quality of service while maintaining lightweight per-client overhead.**

***Index Terms*—volumetric video streaming, synchronization, time-series forecasting, hybrid prediction**

I. Introduction

Volumetric video streaming is emerging as a cornerstone technology for immersive Extended Reality (XR) experiences, enabling real-time interaction with dynamic 3D content such as point clouds, surface meshes, and neural representations. These modalities provide spatial immersion and parallax, which are crucial for telepresence, collaborative virtual environments, and remote training [1], [2]. Unlike conventional 2D streaming, volumetric video imposes significant higher data and computational demands. Even after compression, bitrates can reach hundreds of megabits per second [3], [4]. As XR applications evolve toward scalable, multi-user settings, maintaining tight temporal and spatial synchronization across heterogeneous clients becomes essential. Desynchronization degrades presence and mutual awareness — foundational for effective collaboration in XR — where even small playback or rendering discrepancies can disrupt immersion and cooperative tasks [5], [6].

Achieving synchronization at scale remains a major challenge. Existing approaches, such as timestamp-based alignment, state synchronization, and fixed buffering heuristics, often assume stable network conditions or target low-scale interactions [3], [7]. They struggle under three compounding factors: (i) high variability in network conditions (latency, jitter, packet loss, asymmetric bandwidth, etc); (ii) the need to coordinate delivery across distributed edge–cloud continuum, where varying propagation paths and compute availability affect synchronization; and (iii) the lack of adaptive, predictive synchronization methods that scale efficiently with increasing user count. Optimizing for scalability (via compression, load balancing, or edge computing) often compromises temporal alignment, while enforcing strict synchronization can introduce additional latency and coordination overhead. This trade-off is critical in computationally demanding XR sessions, where ensuring near-simultaneous content delivery quickly becomes a bottleneck for Quality of Service (QoS) and Quality of Experience (QoE) [1], [8], [9].

To address these limitations, this paper presents a predictive, multi-client synchronization framework for scalable volumetric video delivery. The system forecasts per-client latency to anticipate network delays and applies coordinated playback adjustments to maintain alignment while preserving responsiveness. By adapting proactively rather than reacting to lag, the framework reduces inter-client desynchronization, enabling clients to sustain QoS and QoE. Our contributions are as follows:

(1) Predictive synchronization framework: A system that forecasts per-client latency in real time and coordinates playback across distributed clients to maintain temporal alignment under fluctuating network conditions.

(2) Balancing scalability and synchronicity: A coordination strategy that jointly manages playback buffering and synchronization accuracy, minimizing desynchronization without

increasing end-to-end latency.
(3) Coordinated multi-user buffering: A shared buffering mechanism that aligns playback across clients, preserving temporal coherence essential for collaborative XR experiences.
(4) Evaluation under heterogeneous conditions: Validation using real-world network traces across varying client counts and cache sizes, demonstrating improved synchronization stability, robustness, and scalability.

We offer an open-source and reproducible code, which is released at https://github.com/Hellblazer009/XR_Sync/tree/main

II. Background and Related Work

A. XR Architectue and Adaptive Streaming

Recent research has emphasized both architectural considerations and adaptive streaming for immersive XR applications. Asim *et al.* [10] introduced Hera, a modular framework combining application-level streaming logic with delay-based, QoE-aware rate control. Hera adapts dynamically to network variations, maintaining low latency and high video quality, demonstrating the effectiveness of predictive and adaptive mechanisms in multi-user XR contexts. Bentaleb *et al.* [2] provided a comprehensive survey on volumetric video streaming architectures, highlighting the bandwidth and latency challenges of immersive 3D content. The study outlines end-to-end strategies, including encoding, transmission, and adaptation layers, illustrating where predictive or adaptive approaches can mitigate performance bottlenecks. Alsader *et al.* [11] reviewed QoE-driven adaptive streaming toward 6G networks, focusing on architectures and AI techniques that integrate user-perceived quality into rate adaptation and network resource management. This work directly motivates hybrid predictors that combine lightweight continuous models with heavier machine learning models for latency-sensitive streaming.

B. Synchronization-Oriented Approaches

Early work by Kawamura *et al.* [12] synchronized Augmented Reality (AR) content using UTC-based presentation timestamps, effective in single-user contexts but limited in scalability and real-time adaptability. Later studies emphasized multi-user synchronization. Jeon *et al.* [13] proposed *eCAR*, an edge-assisted framework aligning coordinate systems across devices to maintain consistent AR object placement and reduce latency. Similarly, Sonkoly *et al.* [14] developed an edge–cloud coordination platform that offloads SLAM and sensor processing to edge servers, ensuring real-time consistency in shared environments. Beyond system-level coordination, other research examined temporal and interactional synchronization. Thomaschewski *et al.* [15] highlighted synchronization as a collaborative as well as technical challenge, requiring aligned actions and shared context among users. Skorin *et al.* [16] explored XR communication through dynamic digital twins, outlining synchronization challenges between physical and virtual spaces and the need for integrated localization, tracking, and network coordination.

C. Scalability-Oriented Approaches

Recent work has advanced scalability through compression, edge computing, and adaptive streaming. Liu *et al.* [3] addressed scalability in multi-user XR with MuV2, which offloads rendering to edge servers, performs gaze-based view synthesis, and uses encoding multiplexing to reduce redundancy. While effective in bandwidth reduction and responsiveness, it lacks inter-client synchronization for shared, time-sensitive XR sessions. Vivo [17] used tile-based content fetching to lower bandwidth and compute load, but without synchronization mechanisms. Other systems such as Groot [18] and LiveVV [8] enhance scalability through view adaptation and real-time pipelines but assume independent client timelines. Fernandez *et al.* [19] demonstrated scalable holoportation via an MCU-based architecture, supporting more users yet omitting timing alignment. De Fre *et al.* [20] introduced a scalable WebRTC-based volumetric video streaming system using multiple description coding (MDC) and congestion-aware control, achieving low latency but without evaluating synchronization accuracy. Liu *et al.* [21] optimized NeRF-based volumetric streaming for bandwidth and rendering fidelity, while leaving inter-client temporal coherence unaddressed.

III. Methodology

A. System Overview and Scope

We consider a multi-user XR environment where volumetric content is streamed from one or more servers to geographically distributed clients with Head-Mounted Displays (HMDs). The system must preserve co-presence by ensuring near-real-time rendering of shared 3D scenes across users. Two main challenges arise: (i) minimizing inter-client playback skew that disrupts synchronous perception, and (ii) sustaining scalability under heterogeneous and fluctuating network conditions that strain resources. For controlled evaluation, we focus the study to a representative volumetric streaming scenario. Content is transmitted as pre-binarized glTF chunks of fixed size, enabling predictable buffering and synchronization independent of scene complexity. Clients operate using clocks synchronized through standard NTPv4, which typically maintains sub-millisecond offsets on local networks and a few milliseconds over wide-area paths [22]. Given that these offsets are small relative to our synchronization budget, we treat clock drift as negligible in this study. Rendering operations are abstracted to isolate networking and buffering effects, allowing us to focus on the synchronization and coordination layers of the system.

B. System Architecture

Figure 1 illustrates the client–server architecture and data flow. Synchronization is centrally coordinated at the server with lightweight client-side prediction. The server hosts a *video source*, *frame handler*, *buffer controller*, and *heartbeat listener*, while each client runs a *hybrid prediction module*, *heartbeat module*, *buffer listener*, and *frame listener*. Three message types circulate: (1) heartbeat data with measured and predicted latencies $L_i(t)$ and $\hat{L}_i(t)$, (2) buffer commands

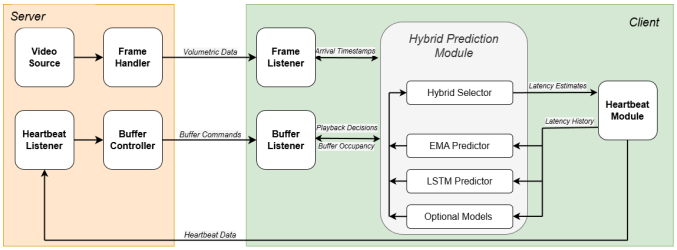

Fig. 1: System architecture and data flow.

adjusting playback buffers $B_i(t)$, and (3) volumetric chunk streams. Clients forecast latency locally, while the server aggregates forecasts to compute global buffer targets, minimizing playback skew. This division distributes lightweight prediction to the clients while retaining centralized control for alignment, sustaining responsiveness as user count increases.

C. Latency Measurement and Clock Synchronization

Our system estimates the *effective* one-way network latency $L_i(t)$ using round-trip time (RTT) bisection with NTP-synchronized clocks and assuming symmetric path delays. Each heartbeat cycle (Fig. 1) proceeds as follows:

1) The client i records the sending timestamp t_s and transmits a heartbeat message containing t_s to the server.
2) The server immediately echoes t_s back to the client.
3) The client receives the response at local time t_r and computes:

$$L_i(t) = \frac{t_r - t_s}{2}.$$

All client and server clocks are synchronized via NTPv4 [23], which maintains sub-millisecond offsets on local-area networks [22]. Given that these offsets are small relative to synchronization budget, we treat clock drift as negligible in this study. For deployments requiring tighter bounds, the Precision Time Protocol (PTP) can achieve sub-microsecond synchronization [24].

System Components: As shown in Figure 1, the system-side components include:

Frame Handler (Server): Streams fixed-size 3D chunks aligned with synchronization commands.

Buffer Controller (Server): Aggregates heartbeat data, computes the maximum predicted latency, and issues buffer commands for synchronized playback.

Heartbeat Listener (Server): Maintains low-bandwidth communication link to the client for latency data.

The client-side components include:

Hybrid Prediction Module (Client): Uses statistical smoothing and lightweight machine learning algorithms (e.g., LSTM) to forecast one-way latency.

Heartbeat Module (Client): Periodically reports measured and predicted latencies.

Buffer Listener (Client): Receives buffer commands from the server and applies buffer adjustments.

Frame Listener (Client): Receives and schedules chunked 3D data for synchronized rendering

Component Interaction and Data Flow: Clients measure latency $L_i(t)$, predict $\hat{L}_i(t)$, and send forecasts to the server. The buffer controller aggregates them, computes global latency bounds, and assigns buffer targets $B_i(t)$ that minimize skew. Commands are returned to clients, which adjust playback while receiving 3D chunks. This feedback loop maintains synchronized playback across distributed clients with minimal overhead.

Architecture Benefits and Comparison: Conventional streaming protocols (e.g., Dynamic Adaptive Streaming over HTTP and HTTP Live Streaming) optimize individual playback but lack inter-client synchronization or predictive latency control. Our framework bridges this gap by combining client-side forecasting with centralized buffering, ensuring a shared temporal reference and consistent QoS. Key benefits include:

- **Forecast-guided buffering:** leveraging latency predictions to proactively adjust playback.
- **Scalability:** centralized buffer computation reduces per-client overhead for for large numbers of users.
- **Flexibility:** supports new predictors, offloading schemes, and hierarchical servers without redesign.

This architecture design is suitable for social XR, collaborative design, and telepresence scenarios that require temporal consistency.

D. Hybrid Latency Prediction

The variability of network conditions and heterogeneity of hardware make single-model predictors inadequate. We employ a hybrid prediction framework that integrates multiple models and dynamically selects the best-performing one. Let $\mathbf{L}_i(t) = [L_i(t-K), \ldots, L_i(t-1)]^\top$ be the last K latency samples for client i. Each predictor P_j outputs a forecast $\hat{L}_i^{(j)}(t)$ and residual

$$r_i^{(j)}(t) = \left| \hat{L}_i^{(j)}(t) - L_i(t) \right|.$$

The *HybridSelector* evaluates recent residuals to choose the most accurate predictor or apply weighted ensembles. Predictors include lightweight models, namely: Exponentially Weighted Moving Average (EWMA) and Autoregressive Integrated Moving Average (ARIMA), and neural models: Long Short-Term Memory neural network (LSTM), balancing responsiveness and long-term accuracy. The modular design also allows plug-and-play integration of newer models (e.g., Prophet, LightGBM, Transformers).

Offline training uses real-world datasets (Kaggle VR [25] and RIPE Atlas [26]) capturing diverse latency patterns. Hyperparameters are tuned via cross-validation; lightweight models adapt continuously, while heavier ones retrain periodically. The hybrid predictor consistently achieves lower residuals and higher real-time accuracy than any single-model baseline while maintaining low overhead.

1) Adaptive Predictor Selection: The hybrid predictor combines greedy residual minimization with threshold-based model activation. Prediction accuracy is measured using Mean Absolute Error (MAE) over a sliding window of $W = 5$

Algorithm 1 Hybrid Greedy Latency Predictor Selection

Require: Latency history $\{L(t-k), \ldots, L(t-1)\}$, window size W

for each heartbeat interval t **do**
  Compute EWMA prediction $\hat{L}_{\text{EWMA}}(t)$
  Compute residual

$$e_{\text{EWMA}}(t-1) = \left| \hat{L}_{\text{EWMA}}(t-1) - L(t-1) \right|$$

  if $\text{MAE}_{\text{EWMA}} > \varepsilon_1$ **then**
    Compute ARIMA prediction $\hat{L}_{\text{ARIMA}}(t)$
    Compute residual $e_{\text{ARIMA}}(t-1)$
  end if
  if $\text{MAE}_{\text{ARIMA}} > \varepsilon_2$ **then**
    Compute LSTM prediction $\hat{L}_{\text{LSTM}}(t)$
    if inference time $> T_{\max}$ **then**
      Discard $\hat{L}_{\text{LSTM}}(t)$
    end if
  end if
  Compute mean residuals over window W
  Select

$$m^\star = \arg\min_{m \in \mathcal{M}_t} \text{MAE}_m$$

  Output $\hat{L}(t) \leftarrow \hat{L}_{m^\star}(t)$
end for

recent samples. EWMA is always active due to constant-time cost. ARIMA activates when EWMA MAE exceeds $\varepsilon_1 = 10\,\text{ms}$, and LSTM activates when ARIMA MAE exceeds $\varepsilon_2 = 20\,\text{ms}$. Among active predictors, the model with lowest MAE is selected greedily. To prevent thrashing, predictor selection occurs once per heartbeat interval ($T_h = 1\,\text{s}$). Once activated, heavier models must remain active for at least W consecutive intervals before de-escalation, providing implicit hysteresis. These thresholds align with designed 20-50 ms QoE synchronization bounds. LSTM inference is bounded by a 20 ms timeout and skipped if exceeded, limiting LSTM to sustained network instability while stable conditions use EWMA or ARIMA with negligible overhead.

E. Playback Adjustment and Buffer Coordination

After computing buffer targets $B_i(t)$, the server issues playback alignment commands. Each client synchronizes playback relative to the slowest predicted latency, ensuring fairness across heterogeneous links. Buffer adjustments are smoothed over short interpolation windows to prevent visible disruption; under sustained degradation, clients can switch servers. By combining predictive latency control with adaptive buffering, the framework achieves near-real-time synchronization across distributed clients without excessive buffering or computational cost.

IV. Experiment & Evaluation

We implement the proposed multi-client volumetric video streaming system in Python using a client–server architecture, where the server streams pre-encoded 3D frames (point clouds) and clients perform hybrid latency prediction and buffer adjustment in real time. We conduct experiments to evaluate scalability, synchronization accuracy, and the impact of prediction models on buffering efficiency.

A. Implementation and Experiment Setup

The server, implemented in Python 3.9 using `asyncio`, streams volumetric video frames at 30 fps over three logical TCP channels: (i) frame streaming, (ii) heartbeat reporting, and (iii) buffer control. Each client runs a lightweight Python application that exchanges timestamped heartbeats with the server to estimate one-way latency, executes the `HybridSelector` combining EWMA, ARIMA, and LSTM predictors, and dynamically adjusts local playback buffers in response to control commands. For evaluation, clients log received frames to measure playback skew and synchronization behavior.

We design three experiments to systematically evaluate:

- **Scalability:** Varying the number of clients from 1 – 150 to analyze how latency, jitter, and server load evolve with concurrency.
- **Chunk Size:** Varying stream chunk sizes (4 – 32 KB) to study cache–latency trade-offs and packet reliability.
- **Prediction Models:** Comparing the hybrid predictor against baselines (EWMA, ARIMA, LSTM, Prophet, and no prediction) under both synthetic and real network traces.

B. Baselines and Evaluation Metrics

We benchmark the proposed hybrid prediction framework against representative synchronization and forecasting methods. Traditional schemes include *state synchronization*, where clients follow global timestamps, and *fixed-frame buffering*, where playback is delayed by a static buffer ($\approx 100\,ms$). Forecasting baselines replace our hybrid module with standalone models, including EWMA, ARIMA, LSTM, and Prophet. Together, these provide a comprehensive comparison across reactive, statistical, and learned predictors.

Evaluation focuses on synchronization quality, buffering efficiency, and scalability:

- **Synchronization Accuracy:** Measured as maximum and average inter-client playback skew (ms).
- **Buffering Efficiency:** Mean and variance of per-client buffer size, indicating adaptation stability.
- **Scalability:** Variation of latency, jitter, and residual prediction error as client count increases.
- **Prediction Accuracy:** Quantified via MAE and RMSE.
- **QoS:** Percentage of frames arriving within 20, 33, and 50 ms playback bounds, representing perceptual synchronization thresholds.

These evaluation metrics allow consistent comparison of synchronization mechanisms under realistic and stress-test conditions, demonstrating how hybrid prediction improves temporal coherence and scalability for multi-user XR streaming.

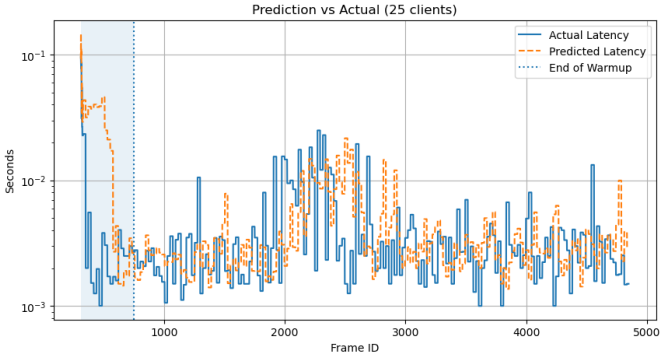

Fig. 2: Prediction results for 25 clients.

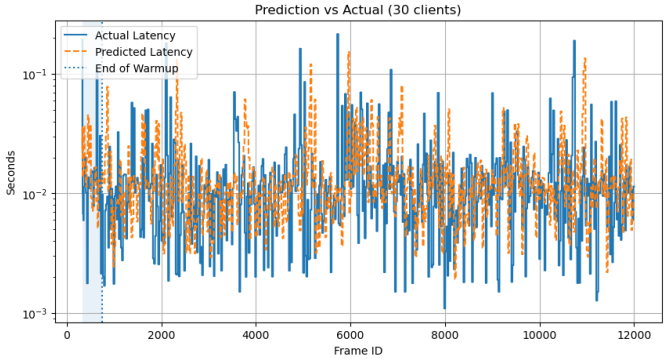

Fig. 3: Prediction results for 30 clients.

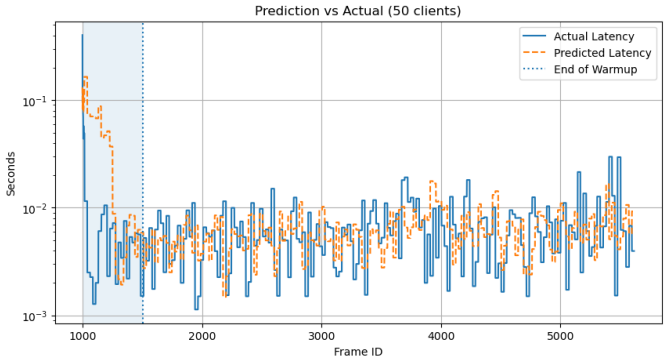

Fig. 4: Prediction results for 50 clients.

V. Results

A. Synchronization Accuracy

We deploy multiple concurrent clients streaming from server. Each client logged its render timestamp $t_i(f)$ for every frame f (at 30 fps). We compute per–frame skew as

$$\text{skew}(f) = \max_i t_i(f) - \min_i t_i(f), \tag{1}$$

and average over all frames in a 5-minute session to obtain the mean skew.

Across all tested conditions, predictive synchronization consistently outperformed the two baselines. Average skew was approximately 18 ms for our method, compared to 52 ms under state synchronization and 76 ms under fixed-frame buffering. This corresponds to a reduction of over 65% in inter-client skew, directly improving the perception of shared real-time experiences. Importantly, the skew achieved by predictive synchronization is below one frame interval (33 ms), meaning that users effectively view frames in lockstep. Figs. 2–4 illustrate prediction residuals for 25, 30, and 50 concurrent clients, respectively. The hybrid predictor maintains stable forecast accuracy across scales, with residuals tightly bounded and no observable drift as the number of clients increases. This consistency confirms that the prediction module generalizes across heterogeneous network paths and load conditions. To ensure that synchronization measurements reflect steady-state behavior, we clearly demarcate the warm-up region on the prediction results (shaded blue in Figs. 2–4). Thus, we exclude startup transients, isolating periods of stable streaming and active synchronization.

QoS compliance analysis (Fig. 5) further highlights this effect, i.e., predictive synchronization maintained over 95% of frames within the 33 ms bound up to 100 clients, whereas both baselines degraded much earlier. These results show that prediction-driven synchronization substantially improves playback alignment in multi-client settings.

B. Buffering Overhead

We record commanded buffer sizes $B_i(t)$ of clients. On average, predictive synchronization reduces buffer requirements compared to fixed buffering (120 ms vs. 200 ms). This 40% reduction demonstrates that proactive prediction enables tighter buffering without sacrificing playback continuity. Smaller buffers not only reduce end-to-end delay but also improve interactivity, a critical property for applications such as volumetric conferencing and shared XR experiences.

Fig. 6 illustrates that the hybrid approach consistently provisions smaller buffers while avoiding the playback stalls observed under rigid fixed buffering. This indicates that accurate latency prediction can serve as a substitute for brute-force buffering, yielding both high efficiency and responsiveness.

C. Scalability Analysis

The scalability experiments highlight two distinct insights. For $N \leq 100$, latency, residual error, and buffer size remain nearly constant (Table I), with mean per-frame latency between 4–20 ms and 95th-percentile latency below 70 ms. Residuals remain low (MAE $\approx$ 5–10 ms), buffer sizes stable (15–80 ms), and QoE compliance above 90% across all tested bounds. This stability suggests that the per-client overhead of synchronization and prediction remains modest even as concurrency scales by two orders of magnitude.

At $N = 150$, however, the system collapses, with average latency exceeding 1 s, residuals approaching 1 s, and QoS compliance dropping to 0%. The observed sharp inflection point suggests a resource saturation limit, arising from CPU and socket scheduling overhead on the single server host, rather than from gradual degradation of the algorithm. The

Fig. 5: QoS impact (fraction of frames that comply with latency requirement.)

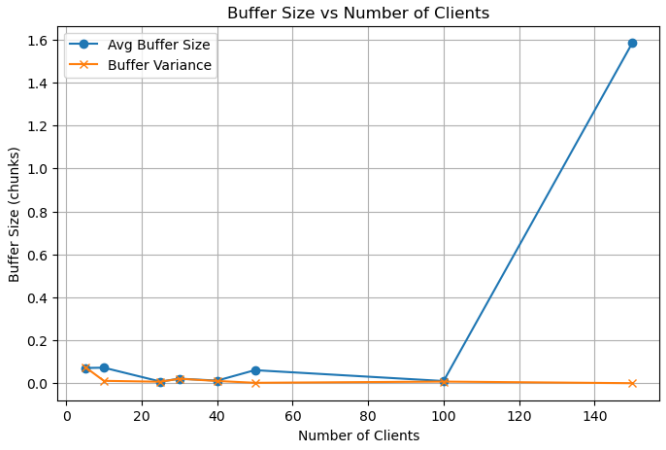

Fig. 6: Average per-client buffer size across methods.

synchronization framework scales effectively to medium-sized deployments; however, supporting more than 100 concurrent clients requires either hardware scaling or a distributed server architecture.

VI. Discussion and Outlook

We contribute a predictive, multi-client synchronization framework for XR applications. Our experiments demonstrate that prediction-driven synchronization offers a practical path toward scalable, low-latency multi-user volumetric streaming. The proposed framework sustains coherent playback across up to 100 clients with stable latency and skew, confirming that the per-client prediction and control overhead is modest. This scalability makes it suitable for collaborative XR use cases such as volumetric conferencing or shared immersive environments without resorting to complex multi-server orchestration.

The hybrid prediction design — combining lightweight estimators (EWMA, ARIMA) with selective escalation to deep models (LSTM) — proves both accurate and efficient. This strategy achieves near-optimal prediction accuracy while avoiding the inference costs of heavy standalone models such as Prophet, underscoring that computational economy is as critical as raw accuracy in real-time systems. The result is reduced buffer occupancy and tighter synchronization, directly improving interactivity and user-perceived QoE.

Clients	Avg Latency (s)	p95 Latency (s)	MAE (s)	QoS $\leq$33ms (%)
30	0.013	0.046	0.011	92.6
50	0.006	0.013	0.007	95.1
100	0.004	0.007	0.004	99.0
150	1.075	1.075	0.950	0.0

TABLE I: Results of scalability metrics for different numbers of clients. The framework sustains stable latency and skew up to 100 clients before server saturation

Beyond these empirical gains, several broader insights emerge. First, hybrid, uncertainty-aware prediction is a viable means to stabilize multi-user playback without excessive buffering. Coordinating models by cost and confidence yields large efficiency gains relative to naïve predictor fusion. Second, scalability challenges stem primarily from system bottlenecks — CPU and socket I/O saturation — rather than synchronization logic. This highlights the importance of observability, admission control, and horizontal scaling when extending to hundreds of clients. Third, end-to-end latency reductions increasingly depend on joint optimization across layers, i.e., static TCP transport and fixed-bitrate volumetric assets limit responsiveness, pointing toward future integration with adaptive-bitrate volumetric codecs and modern transport protocols such as Quick UDP Internet Connections (QUIC) and Web Real-Time Communication (WebRTC).

While these insights are robust under our experimental conditions, they were derived from streaming pre-binarized glTF assets at fixed frame rates to isolate communication effects, with clients co-located and time-synchronized. These choices likely understate clock drift and network asymmetry; nonetheless, NTPv4 typically maintains sub-millisecond offsets on LANs and a few milliseconds over wide-area paths [22] with PTP being even faster, which remains small relative to our synchronization budget. Future evaluations across geographically distributed clients and heterogeneous XR hardware will provide a more complete view of end-to-end performance.

Predictive, group-coordinated buffering emerges as a promising foundation for scalable immersive systems. Distributing synchronization and prediction across multiple servers or edge nodes, and coupling them with adaptive-bitrate volumetric codecs, could further enhance responsiveness under real-world network variability. Open research questions remain on defining temporal consistency for overlapping volumetric scenes, managing fairness across heterogeneous clients, and reconciling privacy with low-latency coordination. Addressing these challenges will be key to advancing intelligent XR architectures from controlled prototypes to robust, global deployments.

Acknowledgment

This work was partially funded by Research Council of Finland InterEarth RESET project, Nordic University Coop-

eration on Edge Intelligence (168043), and Nordic Research Infrastructure Hub for Digital Plant Phenotyping (234087).

REFERENCES

[1] S. Gül, D. Podborski, T. Buchholz, T. Schierl, and C. Hellge, "Low-latency cloud-based volumetric video streaming using head motion prediction," in *Proceedings of the 30th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2020, pp. 27–33.

[2] A. Bentaleb, M. Lim, S. Hammoudi, S. Harous, and R. Zimmermann, "Solutions, challenges, and opportunities in volumetric video streaming: An architectural perspective," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 21, no. 7, pp. 1–35, 2025.

[3] Y. Liu, P. Zhou, Z. Zhang, A. Zhang, B. Han, Z. Li, and F. Qian, "Muv2: Scaling up multi-user mobile volumetric video streaming via content hybridization and sharing," in *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, 2024, pp. 327–341.

[4] A. Zhang, C. Wang, B. Han, and F. Qian, "{YuZu}:{Neural-Enhanced} volumetric video streaming," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 2022, pp. 137–154.

[5] S. Van Damme, J. Sameri, S. Schwarzmann, Q. Wei, R. Trivisonno, F. De Turck, and M. Torres Vega, "Impact of latency on qoe, performance, and collaboration in interactive multi-user virtual reality," *Applied Sciences*, vol. 14, no. 6, p. 2290, 2024.

[6] I. Yeregui, D. Mejías, G. Pacho, R. Viola, J. Astorga, and M. Montagud, "Edge rendering architecture for multiuser xr experiences and e2e performance assessment," in *2024 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. IEEE, 2024, pp. 1–7.

[7] A. Aiersilan and Z. Wang, "A 3d framework for improving low-latency multi-channel live streaming," *arXiv preprint arXiv:2410.16284*, 2024.

[8] K. Hu, Y. Chen, K. Han, B. Li, H. Yang, Y. Jin, J. Liu, and F. Wang, "Livevv: Human-centered live volumetric video streaming system," *IEEE Internet of Things Journal*, 2025.

[9] S. Gül, D. Podborski, J. Son, G. S. Bhullar, T. Buchholz, T. Schierl, and C. Hellge, "Cloud rendering-based volumetric video streaming system for mixed reality services," in *Proceedings of the 11th ACM multimedia systems conference*, 2020, pp. 357–360.

[10] R. Asim, A. Bhardwaj, L. Suramanian, and Y. Zaki, "Towards next generation immersive applications in 5g environments," *arXiv preprint arXiv:2507.20050*, 2025.

[11] M. Alsader, A. A. Barakabitze, and I.-H. Mkwawa, "Qoe-driven adaptive video streaming: Architectures, techniques, and future research challenges toward 6g networks," *IEEE Access*, 2025.

[12] Y. Kawamura, Y. Yamakami, H. Nagata, and K. Imamura, "Real-time streaming of sequential volumetric data for augmented reality synchronized with broadcast video," in *2019 IEEE 9th International Conference on Consumer Electronics (ICCE-Berlin)*. IEEE, 2019, pp. 267–268.

[13] J. Jeon and W. Woo, "ecar: edge-assisted collaborative augmented reality framework," *arXiv preprint arXiv:2405.06872*, 2024.

[14] B. Sonkoly, B. G. Nagy, J. Dóka, Z. Kecskés-Solymosi, J. Czentye, B. Formanek, D. Jocha, and B. P. Gerő, "An edge cloud based coordination platform for multi-user ar applications," *Journal of Network and Systems Management*, vol. 32, no. 2, p. 40, 2024.

[15] L. Thomaschewski, N. Feld, B. Weyers, and A. Kluge, "The use of augmented reality for temporal coordination in everyday work context," in *Everyday virtual and augmented reality*. Springer, 2023, pp. 57–87.

[16] L. Skorin-Kapov, M. Matijasevic, I. Podnar Zarko, M. Kusek, D. Huljenic, V. Skarica, D. Skarica, and A. Grguric, "Shared presence via xr communication and interaction within a dynamically updated digital twin of a smart space: Conceptual framework and research challenges," *Applied Sciences*, vol. 15, no. 16, p. 8838, 2025.

[17] B. Han, Y. Liu, and F. Qian, "Vivo: Visibility-aware mobile volumetric video streaming," in *Proceedings of the 26th annual international conference on mobile computing and networking*, 2020, pp. 1–13.

[18] K. Lee, J. Yi, Y. Lee, S. Choi, and Y. M. Kim, "Groot: A real-time streaming system of high-fidelity volumetric videos," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, 2020, pp. 1–14.

[19] S. Fernandez, M. Montagud, D. Rincón, J. Moragues, and G. Cernigliaro, "Addressing scalability for real-time multiuser holoportation: introducing and assessing a multipoint control unit (mcu) for volumetric video," in *Proceedings of the 31st ACM International Conference on Multimedia*, 2023, pp. 9243–9251.

[20] M. De Fré, J. van der Hooft, T. Wauters, and F. De Turck, "Scalable mdc-based volumetric video delivery for real-time one-to-many webrtc conferencing," in *Proceedings of the 15th ACM multimedia systems conference*, 2024, pp. 121–131.

[21] K. Liu, R. Cheng, N. Wu, and B. Han, "Toward next-generation volumetric video streaming with neural-based content representations," in *Proceedings of the 1st ACM Workshop on Mobile Immersive Computing, Networking, and Systems*, 2023, pp. 199–207.

[22] K. Liang, Z. Yang, W. Fang, and J. Wang, "Performance characterization of network timing with remote traceability via gnss time transfer," *Measurement and Control*, vol. 56, no. 7-8, pp. 1387–1395, 2023.

[23] D. Mills, J. Martin, J. Burbank, and W. Kasch, "Network time protocol version 4: Protocol and algorithms specification," Tech. Rep., 2010.

[24] K. Lee, J. C. Eidson, H. Weibel, and D. Mohl, "Ieee 1588-standard for a precision clock synchronization protocol for networked measurement and control systems," in *Conference on IEEE*, vol. 1588, 2005, p. 2.

[25] Kaggle, "Multi-sensory vr network performance dataset," https://www.kaggle.com/datasets/programmer3/multi-sensory-vr-network-performance-dataset, 2025, [Online].

[26] RIPE, "Ripe atlas measurement data," https://atlas.ripe.net/, 2025, [Online].