Scalable Collocated Multi-User VR Through Virtual Environment User Spatial Coherence

Yuqi Zhou* Purdue University

Shuqi Liao† Purdue University

Yufeng Chen‡ Purdue University

Sonia Fahmy§ Purdue University

Voicu Popescu¶ Purdue University

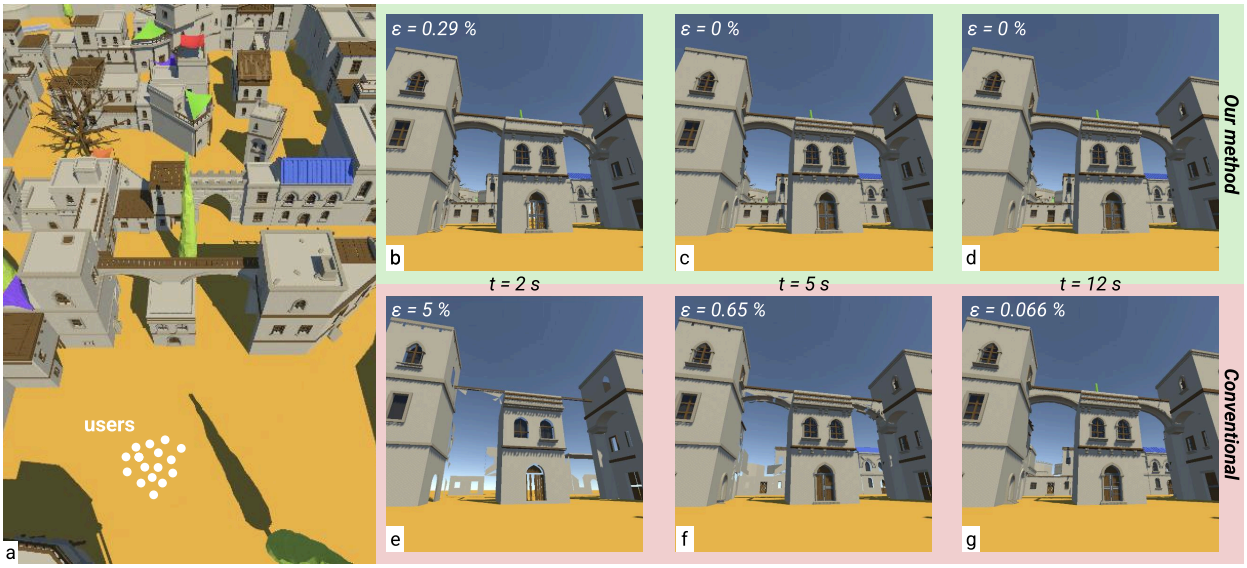

Fig. 1: (a) 16 users collocated in the same real world room start exploring a virtual environment from nearby locations. Our method leverages the user spatial coherence in the virtual environment to send data to all users simultaneously, which leads to faster frame completion (b-d) than for the conventional approach of sending data to each user, one at a time (e-f). With our method, the percentage of incorrect pixels ε in the user frame decreases to zero in 5 s, whereas with the conventional method, incorrect pixels remain even after 12 s. These results correspond to the median user in Figure 9 (right).

Abstract— This paper presents an approach to alleviating the server–client communication bottleneck in collocated multi-user virtual reality (VR). Instead of transmitting unique packets to each client, the system multicasts the same packets to all users, combining packetization, visibility-aware prioritization, and lightweight acknowledgments to ensure efficiency and fairness. To enable rapid scene completion, the environment is partitioned into independent fixed-size packets that can be decoded immediately upon arrival, and transmission order is guided by precomputed visibility footprints. Scalability is achieved through repeated multicast until compact bitmap acknowledgments confirm reception, keeping communication cost mostly independent of the number of clients. A controlled study with 23 participants in groups of 7 and 16 showed that this method reconstructs environments several times faster and with fewer missing parts than a conventional unicast TCP-based approach. These findings demonstrate that multicast transmission over commodity Wi-Fi can support large collocated VR groups.

Index Terms—Multi-user collocated virtual reality, multicast, broadcast, empirical validation.

1 Introduction

Recent advances in virtual reality (VR) technology have enabled quality immersive experiences at commodity prices. All-in-one headsets, such as Meta's Quest 3 [22] and Quest 3S [23], have on-board graphics, tracking, and networking, affording users a completely untethered experience free of external trackers [18]. This accessibility meets the prerequisites for mass deployment across domains like science, engineering, and healthcare. VR is particularly promising for large lectures on university campuses [12]. Education research has well documented that in large lectures (100+ students), student engagement decreases, hindering learning [6]. VR has the potential to reverse this trend by providing students with immersive 3D visualizations of artifacts and allowing instructors to virtually partition the classroom into small, collaborative groups.

However, realizing this potential in a crowded lecture hall faces a critical infrastructure barrier: the **wireless communication bottleneck**. Ideally, a classroom VR system should not require hardware infrastructure beyond the room's standard wireless router and the instructor's laptop. Yet, unlike distributed users who rely on separate Internet connections, collocated users in a single room compete for the same limited wireless spectrum. As the number of clients increases, standard unicast transmission—where the server sends a unique data stream to each user—causes bandwidth saturation and packet colli-

*e-mail: zhou1168@purdue.edu

†e-mail: liao201@purdue.edu

‡e-mail: chen4044@purdue.edu

§e-mail: fahmy@purdue.edu

¶e-mail: popescu@purdue.edu

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

sions, leading to unacceptably high latency and incomplete scene rendering [25].

Addressing this scalability challenge has been a focus of recent networking research, yet a significant gap remains between *theoretical* solutions and *practical* deployment.

- **Physical/MAC Layer Modifications:** Approaches such as SoftCast [13], FlexVi [24], and MuDRA [10] propose modifying the Wi-Fi physical (PHY) or MAC layers to enable efficient multicast. While effective in simulations or Software Defined Radio (SDR) testbeds, these methods rely on customized Wi-Fi drivers or firmware, access to which is unavailable on commercial standalone headsets due to security constraints.

  Additionally, certain systems such as M5 [34] and Badnava et al. [1] leverage millimeter-wave (mmWave) networks and demonstrate high throughput in small-scale multi-user scenarios. However, mmWave is highly susceptible to blockage. In a dense lecture hall, students inevitably occlude each other, breaking the line-of-sight required for stable connections [30].

- **Edge-Cloud Offloading:** Frameworks such as MuVR [17] reduce headset rendering load but typically rely on unicast transmission for the final hop. In a room with 50+ students, the aggregate bandwidth demand exceeds channel capacity regardless of where the rendering occurs.

Consequently, a **deployment gap** exists: theoretical solutions are often undeployable on standard hardware, whereas deployable solutions (standard unicast) fail to scale. Furthermore, unlike video streaming where content is consumed linearly and resolution is easily modulated, VR is an interactive medium where users effectively "teleport" through the scene, creating bursty, unpredictable data demands.

To bridge this gap, we present a scalable multi-user VR system that operates entirely at the **application layer**, ensuring compatibility with commodity off-the-shelf (COTS) hardware without firmware modifications. Our key insight is that collocated users in a shared environment exhibit high **spatial coherence**—they are likely to view similar parts of the scene simultaneously. We exploit this by multicasting geometry packets to all users at once. However, unlike video streaming (which consumes data linearly), interactive VR requires random access to 3D assets. We therefore combine visibility-aware packetization with a novel **lightweight bitmap acknowledgment** mechanism. This allows the server to guarantee reliable delivery for the entire group with $O(1)$ feedback overhead, avoiding the "ACK implosion" typical of reliable multicast.

This paper makes three key contributions:

1. **A Commodity-Compatible Multicast Framework:** We propose the first application-layer multicast protocol for dense 6-DOF VR that bypasses OS-level restrictions, enabling scalable deployment on standard Wi-Fi and locked headsets.

2. **Scalability by Design:** We introduce a prioritized scheduling algorithm driven by group visibility and bitmap ACKs, ensuring that transmission time remains constant regardless of the number of users.

3. **Ecological Validation:** In contrast to simulation-based studies, we evaluate our system in a real-world environment with up to 16 collocated users. Results show that our method reconstructs scenes several times faster and with fewer errors than conventional TCP unicast, validating its robustness in high-interference settings.

2 Related Work

2.1 Multi-User VR: The Bandwidth Bottleneck

Distributed virtual reality has evolved from early theoretical frameworks [26] to complex modern implementations. However, scaling these systems for collocated groups has a number of limitations.

Most platforms rely on unicast transmission, where performance degrades non-linearly when users exceed ten due to channel contention [25]. To address client-side compute limitations, Edge-Cloud systems such as MuVR [17], MuV2 [19], and MuCVR [16] offload rendering to servers. While reducing client GPU load, they paradoxically increase pressure on the network. Similarly, visibility-aware streaming has been used to reduce bandwidth by prioritizing content within a user's view. Systems such as CloudVR [14], Chen et al. [5], and CloVR [35] leverage gaze data to cull invisible sectors. Crucially, both Edge-Cloud and visibility-based approaches treat users in isolation. Even if data is culled, the server still transmits unique streams to each client. In collocated settings where users share a viewing frustum, this results in redundant transmission of identical data, failing to exploit the group's **spatial coherence**.

To overcome bandwidth limits, recent work such as M5 [34] and Badnava et al. [1] utilizes millimeter-wave (mmWave) frequencies. While providing high throughput for small groups, mmWave is brittle in dense setups. As noted in empirical studies [30], human bodies are significant blockers of mmWave signals. In a lecture hall, students inevitably occlude one another, causing frequent link failures that sub-6GHz Wi-Fi (our focus) avoids.

2.2 Group Communication in Networked VR

To address the inefficiency of unicast, researchers have explored multicast solutions. As early as 1995, Macedonia et al. [20] demonstrated IP multicast for scaling distributed environments. However, modern Wi-Fi introduces new reliability challenges.

2.2.1 Physical/Link-Layer Multicast

To tackle the reliability issues of standard Wi-Fi multicast (which lacks ACKs), systems such as SoftCast [13], FlexVi [24], and MuDRA [10] modify the PHY or MAC layers. They employ techniques such as pseudo-analog transmission or leader-based feedback. Other approaches, such as Medusa [28] and M3 [15], utilize "pseudo-multicast" where devices operate in promiscuous mode to overhear unicast transmissions intended for others. While theoretically superior, these approaches require Software Defined Radios (SDR), root-level driver modifications, or promiscuous mode access. Modern commercial headsets (e.g., Meta Quest) operate as "walled gardens" with locked Android kernels, making these modifications difficult to deploy. Our system achieves similar reliability purely at the application layer.

2.2.2 Application-Layer Multicast

Schemes that operate above the driver level are deployable on standard hardware. VRCast [8] and JUMPS [21] use application-layer multicast but focus almost exclusively on **360-degree video**. Video data is linear, allowing simple pre-fetching. In contrast, Xue et al. [32] highlighted how transport choices (TCP vs UDP) impact interactivity in cloud gaming. Chen et al. [3] proposed multicast with view synthesis to reduce bandwidth. However, 6-DOF VR (our focus) differs from video or cloud gaming; it requires random access to 3D geometry as users teleport unpredictably. Techniques designed for linear video streams cannot handle the bursty, dependency-heavy nature of 3D asset streaming.

3 Collocated Multi-User VR System

Our system (Figure 2) is designed to transmit complex virtual environments efficiently to collocated VR users while maintaining fairness, robustness, and scalability. To achieve this, we introduce three main components. First, individual objects are preprocessed into fixed-size packets that can be transmitted, decoded, and rendered independently (subsection 3.1). Second, scene visibility is precomputed offline to enable efficient runtime lookups of packet footprints without costly per-frame calculations (subsection 3.2). Finally, packets are scheduled at runtime using dynamic packet prioritization strategies and bitmap acknowledgment (ACK) to balance quality, completeness, and fairness across clients (subsection 3.3).

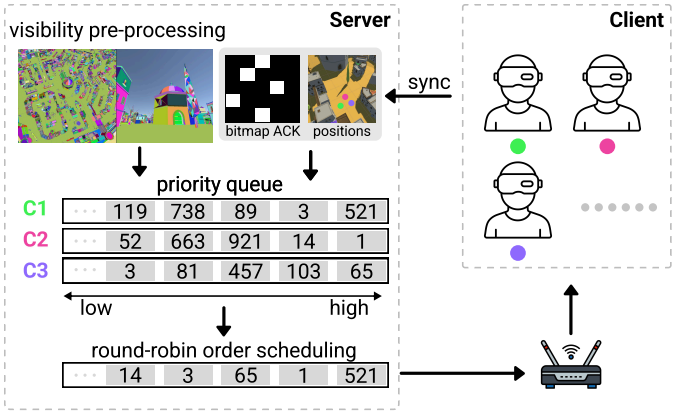

Fig. 2: Overview of the collocated multi-user VR system.

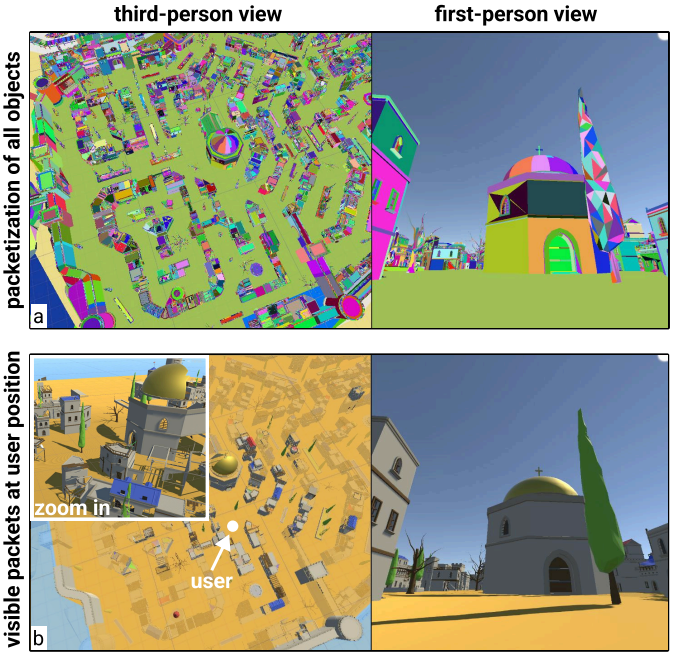

Fig. 3: Object packetization and scene preprocessing. Top row: objects partitioned into fixed-size packets, color-coded by packet. Bottom row: precomputed visibility footprints for a given user viewpoint. Both first- and third-person views are shown.

3.1 Individual Object Packetization

To reduce network transmission, our system transmits only the vertices and triangles visible in the current view, rather than entire objects (Figure 3*b*, zoom in to see the mesh-based effect). Using our partitioning algorithm, transmitting only visible packets cuts the total number of packets by 65–75% compared to sending complete objects, as measured during system implementation.

To avoid the computational overhead of subdividing meshes at runtime—which scales poorly with the number of users—we preprocess each mesh into fixed-size packets using a greedy triangle-grouping strategy (Figure 3*a*). Each object is subdivided into independent packets of uniform size, with each packet containing the object ID, submesh ID, chunk ID, vertex attributes (positions and normals), and triangle indices. This structure allows packets to be decoded and rendered immediately upon arrival, without waiting for the complete object (similar to our work with point clouds [4]). Submeshes and solid-color materials are preserved by recording the corresponding submesh ID, ensuring visual consistency during reconstruction. For texture-based objects, textures are transmitted separately as images, and each packet stores the texture names for reference.

As outlined in Algorithm 1, the procedure begins by selecting a seed triangle and creating a new packet. Subsequent triangles are added to maximize shared vertices with those already included, as long as a packet payload size limit (which we set to 1400 bytes) is not exceeded.

On average, this grouping method increases the number of packets per object by **12.18%** $\pm$ **8.39%** compared to a brute-force split, in which packets contain no shared vertices or triangles. Although brute-force splitting produces slightly fewer packets, it does not guarantee that every packet includes the necessary vertex data for decoding.

3.2 Scene Preprocessing

While computing visible packets directly at runtime guarantees correctness, it is computationally expensive and scales poorly with the number of users. To mitigate this overhead, we preprocess visibility information offline and store it in a structured form for fast online lookup.

The virtual environment (a city-scale scene) is discretized into a regular grid of 1 m $\times$ 1 m cells on the horizontal plane. For each grid cell, we approximate potential user viewpoints by sampling eight positions: the four cell corners at heights of 1 m and 2 m. For each sample, we compute packet footprints using a rasterization-based method. Specifically, the geometry of each packet is rendered into an offscreen buffer, and the number of pixels surviving the depth test is recorded as its visible footprint. This automatically accounts for occlusion, since only fragments passing the depth test contribute to the footprint. The footprint for a given cell is then obtained by aggregating results across the sampled points, yielding a compact representation of the packets likely to be visible and their relative contribution to the final image. Figure 3 illustrates this process across the entire scene, highlighting mesh-based visibility computation from a user's viewpoint to guarantee perceptual completeness.

At runtime, instead of performing costly visibility and projection checks for every user and frame, the server simply identifies the client's current grid cell and retrieves the corresponding precomputed packet footprints. This transforms an otherwise per-frame geometric computation into a lightweight table lookup, greatly reducing server load.

Furthermore, to improve robustness against small head movements that may cross grid boundaries, we expand coverage beyond the current cell by also including visible packets from the surrounding 3$\times$3 neighborhood. If a packet appears in multiple cells within this region, its footprint values are aggregated across those cells when computing its priority.

3.3 Runtime Transmission Scheduling

The server updates each client's priority queue whenever it sends a packet (subsubsection 3.3.1), the client acknowledges received packets (subsubsection 3.3.2), or the client teleports to a new grid cell (subsubsection 3.3.3). Each queue maintains priorities independently per client, as shown in Figure 4.

When transmitting to multiple clients, the server enforces fairness through round-robin scheduling. Because all clients use identical headsets with comparable bandwidth, the server cycles through clients without per-client adaptation. For each client in turn, the server transmits the packet with the largest footprint remaining in that client's queue. Clients with no pending packets are skipped. This mechanism is adapted differently for the conventional method using TCP unicast and for our method using UDP multicast.

In the conventional method, packets are delivered individually to each client, with reliability ensured by TCP's built-in ACK and retransmission mechanisms. For example, as illustrated in Figure 4, the server may first send packet 521 to client C1, then packet 1 to C2, packet 65 to C3, packet 3 to C1, and so on.

In our method, where reliability is not guaranteed, the server repeatedly transmits visible packets until it receives bitmap ACKs from clients confirming successful reception.

Algorithm 1: Preprocessing objects into fixed-size packets

Input: Mesh M with vertices V, normals N, and submeshes $\{S_1, S_2, \ldots, S_k\}$
Input: Maximum packet size $P_{max} = 1400$ bytes
Output: List of packets C
$C \leftarrow \emptyset$;
foreach *submesh* S_i *in* M **do**
  $T \leftarrow$ list of triangles in S_i ;
  $visited \leftarrow \emptyset$;
  while $|visited| < |T|$ **do**
    Create new packet c with metadata (object ID, submesh ID, packet ID, etc.);
    $size \leftarrow$ header size;
    $DIFaddVertices \leftarrow \emptyset$;
    Select an unvisited triangle $t \in T$;
    Add t and its vertices to c ;
    $visited \leftarrow visited \cup \{t\}$;
    Update $size$;
    while *true* **do**
      $best \leftarrow$ triangle $t' \in T \setminus visited$ that maximizes shared vertices with $DIFaddVertices$ and fits within P_{max} ;
      if *no such triangle exists* **then**
        break
      Add t' and new vertices to c ;
      $DIFaddVertices \leftarrow DIFaddVertices \cup \{vectices\ of\ t'\}$;
      $visited \leftarrow visited \cup \{t'\}$;
      Update $size$;
    Append c to C;
return C

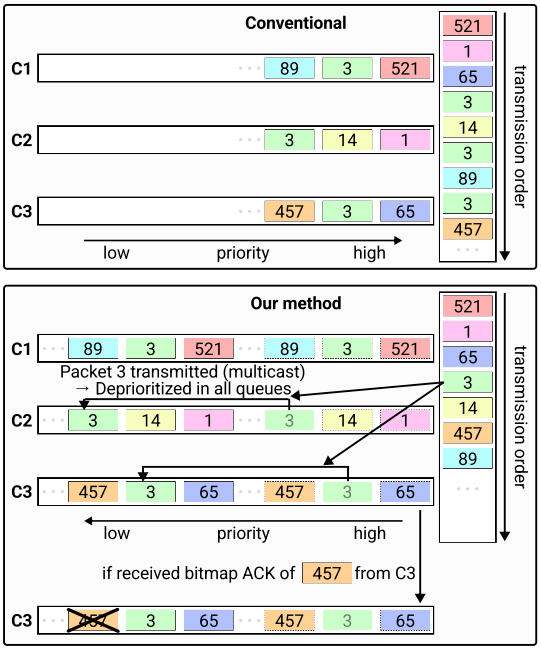

Fig. 4: Dynamic packet prioritization and bitmap acknowledgment. Top: Conventional scheduling (TCP) transmits packets unique to each client sequentially (e.g., Packet 521 to C1, then Packet 1 to C2). Bottom: Our multicast method shares transmissions to reduce redundancy. When the server transmits Packet 3 (green) for Client 1, it is multicast to all users. Since Clients 2 and 3 also receive this data, the server immediately deprioritizes Packet 3 in their queues, moving it from the upcoming high-priority slot (dotted border) to the back of the queue (solid border). Legend: Colors represent unique Packet IDs; identical colors indicate identical data content.

3.3.1 Dynamic Packet Prioritization

For each client, the server tracks how many times each packet (visible or not) has been sent. Packets that have been transmitted fewer times are prioritized over those already sent multiple times, regardless of footprint. For example, in Figure 4, after packet 521 is taken from C1 and sent, it is pushed back in the priority queue even though it has a larger footprint.

This strategy is motivated by two considerations. First, it increases the likelihood that clients reconstruct a complete scene, even if some objects temporarily contain holes due to packet loss, instead of overfocusing on high-footprint packets. Second, because ACKs are returned only every 100 ms (to reduce network congestion and CPU usage), immediately retransmitting frequently sent packets risks unnecessary redundancy. By pushing such packets back in the queue, the server staggers retransmissions, reducing wasted bandwidth and lowering the chance that multiple copies are lost in the same burst.

When a packet is chosen from a client's queue, it is simultaneously sent to all clients via multicast. The server then increments the sending counter for that packet across all clients and pushes it back in their queues if still pending. For instance, in Figure 4, the server first selects packets 521, 1, and 65 from C1, C2, and C3, respectively. When packet 3 is later selected from C1, its counters in C2 and C3 are also incremented, pushing it back in all of their queues. As a result, after taking packet 14 from C2, the server sends packet 457 from C3 instead of 3.

3.3.2 Bitmap Acknowledgment (ACK)

When a client receives a new packet, it needs to inform the server. The primary goal of our bitmap ACK design is not to guide the initial scheduling or transmission of packets, but to eliminate redundant packets during repeated transmissions, enabling the server to focus subsequent transmissions on only the missing ones.

Each client maintains a temporary ACK list of received packet IDs, which it sends to the server every 100 ms and clears after sending. Upon receiving a bitmap ACK, the server removes the acknowledged packets from that client's priority queue. For example, in Figure 4, once packet 457 is sent and acknowledged by C3, the server marks it as received, removes it from C3's queue if present, and excludes it from future scheduling. If a client happens to receive a packet not originally scheduled for them, they will still acknowledge it, ensuring the server does not redundantly assign it to them later.

Compared to traditional retransmission requests, a bitmap ACK is better suited to multicast. Because the server broadcasts the same packet to all clients, per-packet ACKs or NACKs would create excessive feedback traffic. A bitmap ACK instead compresses the state of many packets into a single periodic message, avoiding timers and explicit missing-packet requests. The trade-off is a fixed 100 ms ACK delay, but this is acceptable since clients continue to receive new packets during that interval and dynamic prioritization reduces wasteful retransmissions.

3.3.3 Client Navigation

Packet transmission counts and bitmap ACKs are **not** reset when a client enters a new grid cell. They are only reset upon client reconnection or a full session restart.

Whenever a client enters a new grid cell, its priority queue on the server is cleared and recreated. However, because many packets may already have been received—often thanks to other "pioneer" clients—the actual queue is shorter than it would be under the conventional method. **The more clients move, the more data they share, and the less any single client needs data individually.**

4 Evaluation

We evaluated our method using UDP multicast (experimental condition) against the conventional method using TCP unicast (control condition) through a user study with a session of 16 participants as well as a session of 7 participants. The two sessions were conducted on different days with different participants. The study protocol was reviewed and approved by Purdue University's Institutional Review Board (IRB-2025-1182), which serves as the formal ethical oversight body for human subject research at our institution. The goal was to examine the advantages and limitations of using our approach for larger groups of collocated clients.

4.1 Hypotheses

- *HP1.* For the conventional method, the amount of data transmitted during a time period per client decreases as the number of clients increases. Our method is less affected by the number of clients.

- *HP2.* With our method, clients receive a more complete scene while navigating.

4.2 Setup

The study was conducted in a large classroom on our university campus, with the server and all clients collocated in the same space. Both server- and client-side software were implemented in Unity 6. The server ran on a Windows 11 machine (Dell XPS-8960 [7]) with a 14th-Gen Intel Core i9-14900K CPU (24 cores, 32 threads), 32 GB DDR5 RAM (5600 MT/s), NVIDIA GeForce RTX 4060 Ti (8 GB), and a 1 TB NVMe SSD. Each participant wore a standalone Meta Quest 3 headset [22] and used a right-hand controller.

4.3 Network Configuration

We use a Banana Pi BPI-R3 router [29] running OpenWrt 23.05.2. Our system is evaluated under 802.11a for both unicast and multicast transmissions. Unicast employs the Minstrel rate control algorithm [33], while the multicast (broadcast) physical rate is manually fixed at 54 Mbps. To reduce the impact of power save mode (PSM), we configure the DTIM interval to 1 and set the beacon interval to 10 ms.

4.4 Participants

We recruited a total of 23 participants (13 male, 10 female) with ages ranging from 18 to 28 years ($M = 21, SD = 2.5$). In terms of VR experience, one participant reported never having used a VR headset, five had tried it once, twelve used it occasionally, and five used it frequently.

Pre-experiment eligibility check. Before the participants joined the user study, they read an eligibility questionnaire about whether they were prone to motion sickness, whether they have mobility issues that prevents them from wearing a VR headset, and whether they have normal or corrected vision. After they came to the study, they answered the demographic questionnaire and filled out the consent form. The total time involvement of each participants was less than 60 minutes and was compensated with a gift card.

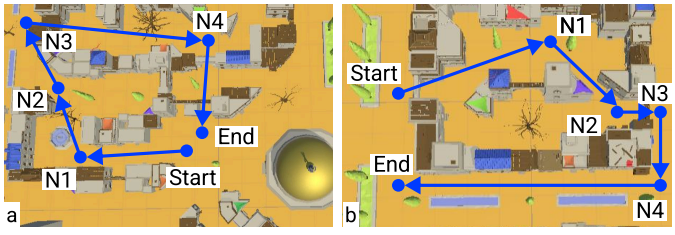

Fig. 5: Two paths used in the user study.

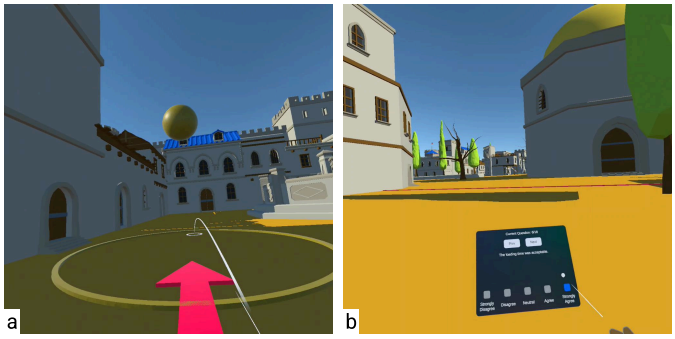

Fig. 6: Experimental procedure. (a) participant teleporting to a node along the path. (b) answering in-VR questions about experience.

4.5 Experiment Design

The experiment exposed participants to both the initial construction of the surrounding environment from an empty scene and the incremental filling of the scene during navigation. Two paths were used, each consisting of a start position, four intermediate nodes, and an end position (Figure 5). The initial data size was 7 MB for the more complex Path 1 and 5 MB for the lighter Path 2, whereas it was 26 MB and 20 MB for full objects before packetization. Each participant experienced both the control and experimental conditions for both paths, resulting in four trials in total (2 paths $\times$ 2 conditions). The trial order was fixed as: Path 1 control, Path 1 experimental, Path 2 control, and Path 2 experimental. This fixed ordering was chosen to standardize the learning curve for VR navigation across all participants, ensuring that any performance differences were due to the method rather than varying familiarity with the controls.

Procedure. Before putting on the headset, participants listened to an explanation of the study flow from the researchers. After launching the application on the headset, they waited until all participants were ready. Each participant initially spawned at the same start position in the virtual environment, seated on the empty ground. At this stage, they could freely rotate their head to look around the scene, but teleportation and thumbstick movement were disabled. To reduce discomfort, thumbstick-based rotation was supported with a peripheral black vignette [9] from Quest SDK.

Once all participants had launched the application, the researchers began sending scene packets. In the control condition, a participant proceeded once they had received all required packets. In the experimental condition, a participant proceeded once the server had transmitted the necessary packets three times. In both cases, the server notified participants when they could start answering questions about their experience while seated in the virtual environment. Figure 6 *right* shows the view of answering questions.

After answering the questions, the first path node (N1) appeared and participants began moving along the path using teleportation. When a participant arrived at a path node by teleporting in the circle (Figure 6 *left*), the current node disappeared and the next node appeared. Each node was visible from the previous one, and an arrow was always pointing from the participant's current position to the next node until they reached the end point. Upon reaching the end position, participants again could not move and answered the same set of questions. Those who finished early were allowed to remove the headset and rest until others completed the task.

After all participants reached the end position and finished the questionnaire, the researcher instructed them to put the headset back on. The server then moved participants to the new start position, removed all objects in the client view, cleared transmission counts and bitmap ACKs, and initialized the next round.

4.6 Data Collection

The client headset recorded the log of position and orientation, as well as the packets received each frame. This data allows simulating the

movement offline and calculating the metrics for evaluating user experiences. Other than the objective data log, we also asked each participant a few questions (8 custom questions + 10 SUS [2]) after the initial transmission and after the navigation for each path. The custom questions are on a five-point Likert scale:

1. I noticed many missing parts in the scene while it was loading.
2. The scene became engaging quickly, even though it was still loading.
3. I noticed many holes in the scene while it was loading.
4. The scene never loaded completely.
5. The parts of the scene appearing as they were loaded were distracting.
6. The loading time was acceptable.
7. I would prefer to see a blank screen until the entire scene is loaded.
8. I like seeing the scene fill in gradually.

4.7 Data Analysis

We analyze the following quantitative metrics over time based on the log files recorded from the client headsets. For each frame recorded, we read the position and orientation so that we can know and locally load the ground truth of what the client can see. We also recorded the accumulated received packets along the path.

- missed packets percentage: the percentage of necessary packets at the current position received by the participants.
- pixel level depth error: the percentage of correct pixels in terms of depth on the client view, compared to the ground truth with all full objects.

We used a Wilcoxon signed-rank test [31] with Holm-corrected p-value [11] and Rosenthal's r value [27] to find out the significance level of both quantitative and qualitative between the control and experimental conditions.

4.8 Results

4.8.1 Quantitative Results

Percentage of missed packets over time. Figure 7 shows the percentage of missed packets over time along the four paths for the 16 participants session. At the beginning of each path, when participants remained stationary, all curves exhibited similar slopes and dropped to 0% within a comparable time window. As summarized in Table 1, our method allowed participants to recover packets much faster: on average, 95% of packets were received by 4.4 s and 99% by 4.7 s, compared to 48.8 s and 51.5 s for the conventional method. Significance tests confirmed these differences at both 5% and 1% thresholds across all paths ($p < 0.001$, $r > 0.87$, very large effect).

Results for the 7-participant session (Table 2) showed the same pattern: our method consistently outperformed the conventional method ($p < 0.001$, $r > 0.87$, very large effect). However, average times for the conventional method were nearly halved relative to the 16-participant session, supporting *HP1* that conventional performance depends on the number of clients. By contrast, recovery times for our method remained stable, with minor differences attributable to UDP variability across sessions ($p > 0.05$), further supporting *HP1*.

After participants began navigating, packet missing diverged sharply between methods. With the conventional method, all participants experienced severe missing packets while moving (left panels of Figure 7). With our method, only the first participant to advance suffered noticeable missing packets; subsequent participants received their required packets through multicast, keeping their missing rate below 25%. Averaged results (Figure 8) confirm this trend even with fewer participants. This supports *HP2*, showing that our method reduces packets missing during movement by leveraging shared transmission.

Pixel level depth error. Figure 9 shows the individual depth error traces over time for the 16-participant session. At the beginning, when all participants remained stationary, the conventional method produced a smooth and rapid decrease in error, reflecting the advantage of a reliable channel in transmitting packets with large footprints. With our method, each packet is equally prone to being missing; in the worst case, participants may miss packets corresponding to large visible areas, resulting in higher depth errors for one or two individuals (see zoomed-in regions of the top-right and bottom-right panels). Thanks to bitmap ACKs, however, these missing packets were quickly recovered, and errors converged.

Figure 10 shows the average depth errors for the 16- and 7-participant sessions. The sharp initial drop illustrates the sending priority strategy, which favors unsent packets over those with large footprints. In the 16-participant session, our method showed slightly higher initial errors than the conventional method but rapidly converged to zero after repeated transmissions. The comparison across group sizes supports *HP1*: for the conventional method, depth error decreased faster with 7 participants than with 16, while for our method the decrease was stable or slightly slower, showing robustness to group size changes.

During navigation, our method further outperformed the conventional approach. As shown in Figure 9, participants who followed the "pioneer" on a path exhibited minimal depth errors, while only the pioneers experienced occasional spikes. In the averaged results (Figure 10), spikes under our method were fewer and smaller compared to the conventional method. This provides additional support for *HP2* in terms of depth error.

Table 1: **N=16**. Time [s] statistics for packet loss rate at thresholds $T = 5\%$ and $T = 1\%$.

Path	Method	T	Mean	Median	Min	Max
Path A	Conventional	5%	48.84	48.80	48.35	49.72
		1%	51.48	51.58	50.31	53.25
	Our method	5%	4.40	4.42	4.28	4.53
		1%	4.71	4.71	4.58	5.24
Path B	Conventional	5%	30.23	30.05	30.02	31.68
		1%	31.45	31.26	31.22	32.88
	Our method	5%	2.26	2.26	1.92	3.07
		1%	3.14	3.10	2.96	3.60
Path A+B	Conventional	5%	39.54	40.01	30.02	49.72
		1%	41.47	41.59	31.22	53.25
	Our method	5%	3.33	3.67	1.92	4.53
		1%	3.93	4.09	2.96	5.24

Table 2: **N=7**. Time [s] statistics for packet loss rate at thresholds $T = 5\%$ and $T = 1\%$.

Path	Method	T	Mean	Median	Min	Max
Path A	Conventional	5%	25.94	25.92	25.75	26.29
		1%	27.10	27.01	26.79	27.34
	Our method	5%	5.09	4.87	4.82	6.46
		1%	5.88	5.72	5.37	6.67
Path B	Conventional	5%	16.05	15.94	15.66	16.51
		1%	16.46	16.48	16.25	16.68
	Our method	5%	3.61	3.26	2.96	5.58
		1%	4.21	3.80	3.56	5.83
Path A+B	Conventional	5%	20.99	21.13	15.66	26.29
		1%	21.78	21.73	16.25	27.34
	Our method	5%	4.35	4.82	2.96	6.46
		1%	5.04	5.49	3.56	6.67

4.8.2 Qualitative Results

Custom questions. For the **initial** stationary stage of the 16 participants session (Figure 11 *top*), the Wilcoxon signed-rank tests ($N = 16$) showed that for **Path A**, no significant differences were found for Q1–Q5 and Q7–Q8 ($p_{\text{holm}} \geq 0.07$), although Q5 (distraction) approached significance ($Z = -2.0$, $p_{\text{holm}} = 0.07$, $r = 0.69$, large). A significant difference was observed only for Q6 (loading time), with a very large effect ($Z = -2.5$, $p_{\text{holm}} = 0.01$, $r = 0.76$). For **Path B**,

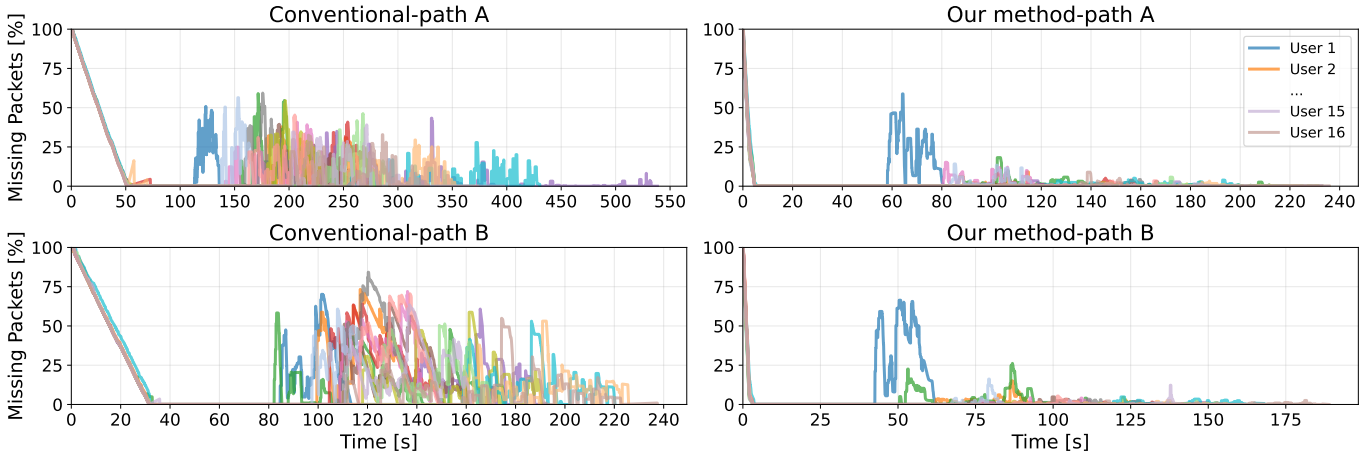

Fig. 7: Missing packets over time for participants across two paths (A, B)

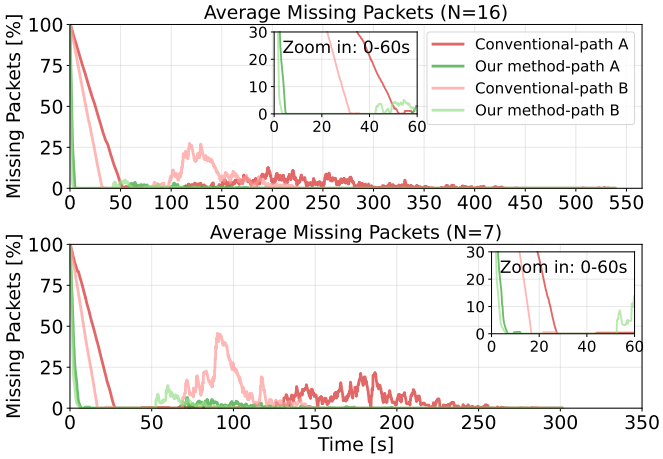

Fig. 8: Average percentage of missing packets for participants across both paths. Top: **N=16**; bottom: **N=7**.

significant differences emerged for Q1 (missing parts: $Z = -2.5$, $p_{\text{holm}} = 0.02$, $r = 0.66$, large), Q3 (holes: $Z = -2.8$, $p_{\text{holm}} = 0.01$, $r = 0.89$, very large), and Q6 (loading time: $Z = -2.8$, $p_{\text{holm}} = 0.01$, $r = 0.89$, very large). Other items on Path B did not reach significance ($p_{\text{holm}} \geq 0.08$), though several showed large to very large effect sizes (e.g., Q5: $r = 0.69$; Q7: $r = 0.77$; Q8: $r = 0.95$). For the **combined analysis across Paths A and B**, significant differences were observed for Q1 (missing parts: $Z = -2.4$, $p_{\text{holm}} = 0.03$, $r = 0.50$, medium), Q5 (distraction: $Z = -2.5$, $p_{\text{holm}} = 0.02$, $r = 0.68$, large), and Q6 (loading time: $Z = -3.8$, $p_{\text{holm}} < 0.01$, $r = 0.84$, very large). The remaining questions did not yield significant differences ($p_{\text{holm}} \geq 0.06$).

For the **moving stage** (Figure 11 *bottom*), the Wilcoxon signed-rank tests ($N = 16$) showed no significant differences across any items on **Path A** ($p_{\text{holm}} \geq 0.14$), although moderate to large effect sizes were observed for Q4 (scene never loaded completely: $Z = -1.6$, $p_{\text{holm}} = 0.14$, $r = 0.64$) and Q7 (preference for blank screen: $Z = -0.8$, $p_{\text{holm}} = 0.41$, $r = 0.46$). In contrast, several items reached significance on **Path B**: Q1 (missing parts: $Z = -3.2$, $p_{\text{holm}} < 0.01$, $r = 0.88$, very large), Q2 (quick engagement: $Z = -2.5$, $p_{\text{holm}} = 0.03$, $r = 0.89$, very large), Q3 (holes: $Z = -3.1$, $p_{\text{holm}} < 0.01$, $r = 0.88$, very large), Q5 (distraction: $Z = -2.6$, $p_{\text{holm}} = 0.03$, $r = 0.82$, very large), and Q6 (loading time: $Z = -2.9$, $p_{\text{holm}} = 0.01$, $r = 0.89$, very large). Q7 and Q8 also showed large effects ($r = 0.72$ and 0.74, respectively) but did not reach significance after correction. For the **combined analysis across Paths A and B**, significant improvements were observed for Q1 ($Z = -3.4$, $p_{\text{holm}} < 0.01$, $r = 0.68$, large), Q2 ($Z = -2.4$, $p_{\text{holm}} = 0.03$, $r = 0.62$, large), Q3 ($Z = -3.2$, $p_{\text{holm}} < 0.01$, $r = 0.67$, large), Q4 ($Z = -2.4$, $p_{\text{holm}} = 0.04$, $r = 0.68$, large), Q5 ($Z = -2.5$, $p_{\text{holm}} = 0.03$, $r = 0.58$, large), and Q6 ($Z = -3.2$, $p_{\text{holm}} < 0.01$, $r = 0.79$, very large), while Q7 and Q8 again showed large but non-significant effects ($p_{\text{holm}} \geq 0.08$).

When comparing to the **initial stationary stage**, where significant differences were limited primarily to Q6 (loading time) and selected perceptual items on Path B, the moving stage produced a broader set of significant improvements, particularly on Path B and in the combined analysis. This pattern suggests that the benefits of our method became more salient during active navigation, not only in perceived loading time but also in reducing missing content, holes, and distraction, and in fostering quicker engagement with the scene, supporting *HP2* in a subjective way.

For the **7-participant session** ($N = 7$) (Figure 12), no significant differences emerged between the conventional method and ours across any questions or paths after Holm correction. Most items had $p_{\text{holm}} \geq 0.09$, though several showed large to very large effect sizes despite nonsignificance (e.g., Path A Q1: $Z = 2.2$, $p_{\text{holm}} = 0.09$, $r = 0.90$; Path B Q3: $Z = -1.6$, $p_{\text{holm}} = 0.75$, $r = 0.93$). A similar pattern was observed in the moving stage: no items reached significance ($p_{\text{holm}} \geq 0.14$), yet effect sizes remained large (e.g., Path A Q3: $Z = -1.3$, $p_{\text{holm}} = 0.50$, $r = 0.95$; Path B Q1: $Z = -1.6$, $p_{\text{holm}} = 0.50$, $r = 0.93$).

In contrast, the **16-participant session** revealed multiple significant differences, especially during the moving stage, where our method substantially reduced perceptions of missing content, holes, distraction, and loading time on Path B and in the combined analysis. The discrepancy between the $N = 7$ and $N = 16$ cohorts reflects the role of statistical power in detecting effects. More importantly, the consistent pattern supports *HP1*: with the conventional method, participants' perceptions degraded as client count increased, consistent with reduced effective bandwidth per client. By comparison, our method was more robust under higher concurrency, maintaining similar perceptual ratings even with 16 simultaneous users.

SUS scores. For the **16-participant sessions** (Figure 13, top), SUS scores were consistently higher for our method. In the **initial stage**, significant differences were found on Path B ($p_{\text{Holm}} = 0.04$) and Path A+B ($p_{\text{Holm}} = 0.04$), while Path A showed no difference. In the **moving stage**, scores again favored our method, though without significance ($p_{\text{Holm}} \geq 0.10$). Our method generally reached or exceeded the SUS threshold of 80 (good usability), whereas the conventional method remained closer to the marginal threshold of 68. For the **7-participant session** (Figure 13, bottom), no significant differences were detected, and both methods remained near the marginal usability range.

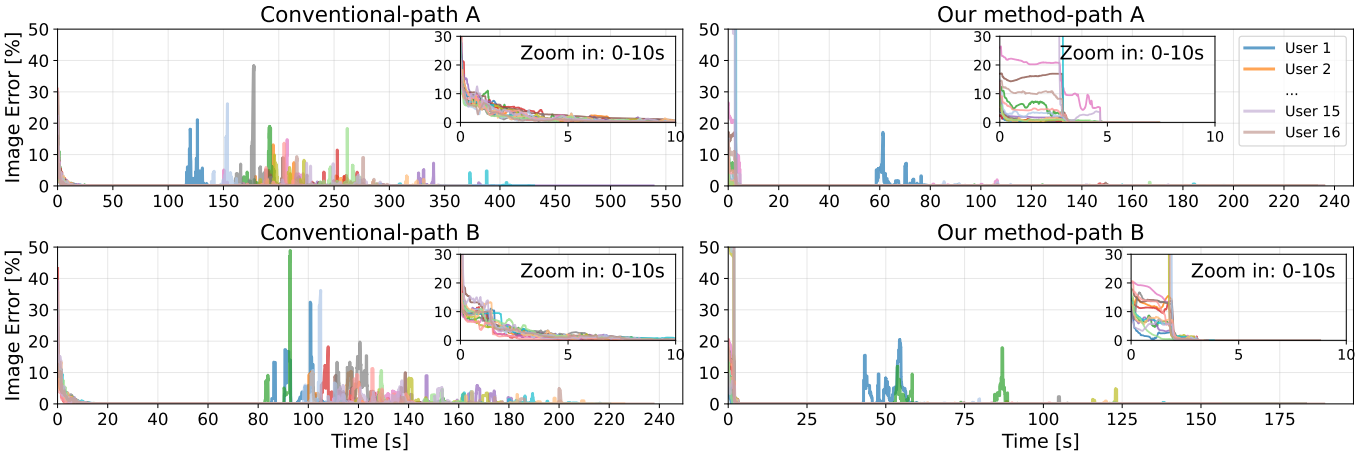

Fig. 9: **N=16.** Pixel-level depth error for individual participants.

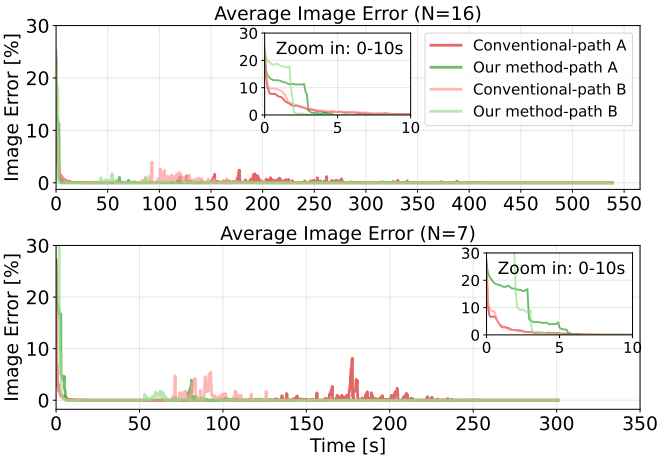

Fig. 10: Average pixel-level depth error for participants. Top: **N=16**; bottom: **N=7**.

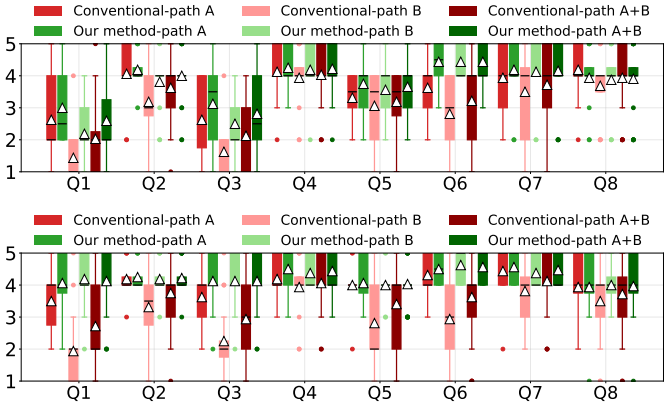

Fig. 11: **N=16.** Top: **initial loading stage**; bottom: **moving stage**. Custom questionnaire results in boxplots across groups A, B, and A+B. White triangles indicate means. Negatively worded items are reverse-scored (higher = better).

5 Discussion

The results of our evaluation demonstrate that our method effectively breaks the linear dependency between user count and scene transmission time, achieving near-constant scene recovery times ($O(1)$) regardless of group size. Beyond these quantitative metrics, the findings offer broader insights into the trade-offs required to design deployable multi-user VR systems.

5.1 Bridging the Deployment Gap

A significant challenge in networked VR is the disconnect between theoretical efficiency and practical deployability. Prior research has demonstrated that Physical/MAC-layer multicast schemes (e.g., [10][24]) can theoretically optimize transmission rates. While these approaches offer superior raw throughput, they suffer from a critical **deployment gap**: they typically require root-level driver modifications or specialized Software Defined Radios (SDRs) [34]. Such access is strictly prohibited on modern commercial headsets (e.g., Meta Quest series), which operate as secure, locked appliances.

Our work addresses this gap by shifting the reliability logic to the application layer. By implementing a lightweight bitmap acknowledgment mechanism over standard UDP, our method achieves the *functional benefits* of robust multicast—scalability without violating the "walled garden" constraints of commodity hardware. This validates that scalable, dense VR is achievable *now* with off-the-shelf infrastructure, rather than being contingent on future open-hardware standards.

5.2 Generalizability and Environmental Constraints

Our evaluation focused on a single, highly collocated environment, which represents the **worst-case scenario** for wireless contention. In a classroom setting, dozens of devices compete for the same bandwidth within a confined physical space, maximizing interference and noise floor. Since our system proved robust under these high-interference conditions ($N = 16$ on a single access point), we anticipate performance would arguably improve in more distributed or spatially sparse setups where signal-to-noise ratios are more favorable.

However, the efficacy of our approach is inextricably linked to **Virtual Spatial Coherence**—the assumption that users view overlapping content. In educational contexts (e.g., a geology class examining a canyon) or guided tours, this assumption holds true. In scenarios with high divergence (e.g., 50 users exploring 50 distinct virtual worlds independently), the benefits of multicast would diminish, and the system would revert to the efficiency of unicast. Future implementations could address this by adopting a *hybrid scheduling* approach, dynamically switching between multicast for the "main group" and unicast for "explorers" based on real-time clustering analysis.

5.3 Ecological Validity and User Experience

While our technical metrics demonstrate bandwidth efficiency, the subjective results provide a critical validation of **ecological viability**. In VR, transmission latency is not merely a performance statistic; it translates directly to motion sickness and breaks in presence—factors that simulation-based studies cannot capture.

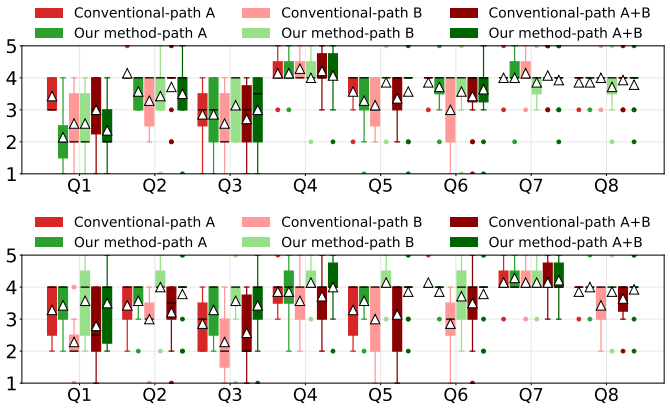

Fig. 12: **N=7.** Top: **initial loading stage**; bottom: **moving stage**. Custom questionnaire results in boxplots across groups A, B, and A+B. White triangles indicate means. Negative items are inverted (higher = better).

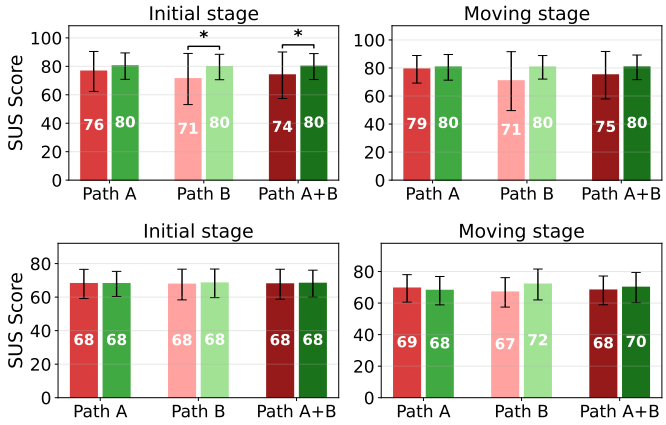

Fig. 13: **SUS usability scores.** Top: **N=16**; bottom: **N=7**.

Our results demonstrate that the technical stability of application-layer multicast ($O(1)$ scaling) **effectively translates** to perceptual comfort. The statistically significant reduction in "distraction" and "perceived holes" confirms that our packetization strategy satisfies the human perceptual threshold for continuity. This corroborates our technical findings, proving that the system's bandwidth savings do not come at the cost of the user's physiological well-being or immersion.

6 Conclusion, Limitations, and Future Work

This paper demonstrated that application-layer multicast, combined with visibility-aware scheduling and lightweight acknowledgments, enables collocated VR users to load and navigate complex environments significantly faster than conventional unicast TCP. By shifting reliability logic from the kernel to the application, we bridge the gap between theoretical network efficiency and practically deployable commodity hardware. A controlled study with 23 participants confirmed that this approach scales efficiently ($O(1)$) where per-client transmission degrades ($O(N)$), while maintaining a quality of experience that users find physiologically comfortable.

Limitations. Our primary constraint is the reliance on **Virtual Spatial Coherence**—the assumption that users view overlapping content. While this holds true for educational contexts (e.g., guided tours, lectures), the benefits of multicast diminish in scenarios with high divergence, such as users exploring distinct virtual worlds independently. Additionally, our evaluation prioritized visual completeness and perceived stability; further longitudinal studies are needed to quantify the impact of multicast latency on highly interactive, fine-motor tasks.

Future Work. To address the coherence limitation, future work will explore **hybrid scheduling** that dynamically switches between multicast for the main group and unicast for divergent "explorers." We also plan to extend the evaluation to multi-AP deployments in larger lecture halls, testing the system's resilience under variable signal-to-noise ratios and heterogeneous device capabilities. These steps will move collocated VR closer to ubiquitous deployment at lecture scale while preserving the efficiency required for high-fidelity immersion.

Acknowledgments

We thank Umakant Kulkarni, Akhil Prasad, and Nathaniel Cherian, as well as the members of the Purdue Computer Science Extended Reality Laboratory (XR Lab), for their feedback. This material is based upon work supported by the United States National Science Foundation under Awards No. 2212200, 2318657, 2309564, 2506783, and 2417510.

References

[1] B. Badnava, J. Chakareski, and M. Hashemi. Neural-enhanced rate adaptation and computation distribution for emerging mmwave multi-user 3d video streaming systems. *IEEE Transactions on Multimedia*, (99):1–12, 2025. 2

[2] J. Brooke. SUS-A quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996. 6

[3] C.-Y. Chen, C.-H. Wang, S.-H. Chiang, D.-N. Yang, and W. Liao. Multicast with view synthesis for wireless virtual reality. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1–7. IEEE, 2020. 2

[4] Y. Chen, U. Kulkarni, V. Popescu, and S. Fahmy. RUN: A case for cross-layer networked virtual reality. In *Proceedings of the 33rd ACM International Conference on Multimedia*, MM '25, p. 12111–12120. Association for Computing Machinery, New York, NY, USA, 2025. doi: 10.1145/3746027.3754800 3

[5] Y. Chen, H. Kwon, H. Inaltekin, and M. Gorlatova. VR viewport pose model for quantifying and exploiting frame correlations. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, pp. 1269–1278. IEEE, 2022. 2

[6] J. Cuseo. The empirical case against large class size: Adverse effects on the teaching, learning, and retention of first-year students. *The Journal of Faculty Development*, 21(1):5–21, 2007. 1

[7] Dell Technologies. Dell XPS 8960 desktop. https://www.dell.com/en-us/shop/reviewselections/xps-8960-desktop/usexpsthcto8960rpl27?configurationId=6dd446c2-7d63-411a-aa06-dbdaaf1e53d2, 2025. Accessed: 2025-09-12. 5

[8] O. Eltobgy, O. Arafa, and M. Hefeeda. Mobile streaming of live 360-degree videos. *IEEE Transactions on Multimedia*, 22(12):3139–3152, 2020. 2

[9] A. S. Fernandes and S. K. Feiner. Combating VR sickness through subtle dynamic field-of-view modification. In *2016 IEEE symposium on 3D user interfaces (3DUI)*, pp. 201–210. IEEE, 2016. 5

[10] V. Gupta, C. Gutterman, Y. Bejerano, and G. Zussman. Experimental evaluation of large scale wifi multicast rate control. *IEEE Transactions on Wireless Communications*, 17(4):2319–2332, 2018. 2, 8

[11] S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pp. 65–70, 1979. 6

[12] A. Ismayilzada, A. Karimov, and M. Saarela. Scaling virtual classrooms: Overcoming barriers to learning analytics in VR. In *CEUR Workshop Proceedings*. RWTH Aachen, 2025. 1

[13] S. Jakubczak and D. Katabi. Softcast: One-size-fits-all wireless video. In *Proceedings of the ACM SIGCOMM 2010 conference*, pp. 449–450, 2010. 2

[14] T. Kämäräinen, M. Siekkinen, J. Eerikäinen, and A. Ylä-Jääski. Cloudvr: Cloud accelerated interactive mobile virtual reality. In *Proceedings of the 26th ACM international conference on Multimedia*, pp. 1181–1189, 2018. 2

[15] M. Li, D. Pei, X. Zhang, B. Zhang, Z. Wang, H. Xu, and Z. Wang. M 3: Practical and reliable multi-layer video multicast over multi-rate wi-fi network. In *2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*, pp. 1–10. IEEE, 2016. 2

[16] W. Li, Q. Li, W. Tian, J. Gao, F. Wu, J. Liu, and J. Ren. MUCVR: Edge computing-enabled high-quality multi-user collaboration for interactive MVR. *IEEE Transactions on Parallel and Distributed Systems*, 2025. 2

[17] Y. Li and W. Gao. Muvr: Supporting multi-user mobile virtual reality with resource constrained edge cloud. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, pp. 1–16. IEEE, 2018. 2

[18] S. Liao, Y. Zhou, and V. Popescu. Ar interfaces for disocclusion—a comparative study. In *2023 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*, pp. 530–540. IEEE, 2023. 1

[19] 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*, pp. 327–341, 2024. 2

[20] M. R. Macedomia, M. J. Zyda, D. R. Pratt, D. P. Brutzman, and P. T. Barham. Exploiting reality with multicast groups: a network architecture for large-scale virtual environments. In *Proceedings Virtual Reality Annual International Symposium'95*, pp. 2–10. IEEE, 1995. 2

[21] A. Majidi and A. H. Zahran. Optimized joint unicast-multicast panoramic video streaming in cellular networks. In *2020 IEEE 28th International Conference on Network Protocols (ICNP)*, pp. 1–6. IEEE, 2020. 2

[22] Meta. Quest 3 Mixed Reality Headset. https://www.meta.com/quest/quest-3/. Accessed: 2024-10-02. 1, 5

[23] Meta. Quest 3S Mixed Reality Headset. https://www.meta.com/quest/quest-3s/. Accessed: 2024-10-02. 1

[24] Y. D. Park, S. Jeon, J.-P. Jeong, and Y.-J. Suh. Flexvi: Phy aided flexible multicast for video streaming over ieee 802.11 wlans. *IEEE Transactions on Mobile Computing*, 19(10):2299–2315, 2019. 2, 8

[25] V. Parthasarathy, A. A. Simiscuka, N. O'Connor, and G.-M. Muntean. Performance evaluation of a multi-user virtual reality platform. In *2020 International Wireless Communications and Mobile Computing (IWCMC)*, pp. 934–939. IEEE, 2020. 2

[26] B. Roehl. Distributed virtual reality: an overview. In *Proceedings of the first symposium on Virtual reality modeling language*, pp. 39–43, 1995. 2

[27] R. Rosenthal. Meta-analytic procedures for social science research sage publications: Beverly hills, 1984, 148 pp. *Educational Researcher*, 15(8):18–20, 1986. 6

[28] S. Sen, N. K. Madabhushi, and S. Banerjee. Scalable wifi media delivery through adaptive broadcasts. In *NSDI*, pp. 191–204, 2010. 2

[29] Sinovoip / Banana Pi Project. Banana Pi BPI-R3 Router Board. Banana Pi Wiki — "Getting Started with BPI-R3", 2024. Spec: MediaTek MT7986 (Filogic 830) quad-core ARM Cortex-A53, 2 GB DDR, etc. 5

[30] C. Slezak, V. Semkin, S. Andreev, Y. Koucheryavy, and S. Rangan. Empirical effects of dynamic human-body blockage in 60 GHz communications. *IEEE Communications Magazine*, 56(12):60–66, 2018. 2

[31] F. Wilcoxon. Individual comparisons by ranking methods. In *Breakthroughs in Statistics: Methodology and Distribution*, pp. 196–202. Springer, 1992. 6

[32] Z. Xue, D. Wu, J. He, X. Hei, and Y. Liu. Playing high-end video games in the cloud: A measurement study. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(12):2013–2025, 2014. 2

[33] W. Yin, P. Hu, J. Indulska, K. Białkowski, et al. On the performance of rate control algorithm minstrel. In *Proceedings of the 2012 IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 406–412, 2012. doi: 10.1109/PIMRC.2012.6362482 5

[34] D. Zhang, P. Zhou, B. Han, and P. Pathak. M5: Facilitating multi-user volumetric content delivery with multi-lobe multicast over mmWave. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*, SenSys '22, p. 31–46. Association for Computing Machinery, New York, NY, USA, 2023. doi: 10.1145/3560905.3568540 2, 8

[35] Y. Zhou and V. Popescu. Clovr: Fast-startup low-latency cloud vr. *IEEE Transactions on Visualization and Computer Graphics*, 30(5):2337–2346, 2024. doi: 10.1109/TVCG.2024.3372059 2