

CRDT-Based Game State Synchronization in Peer-to-Peer VR

Abel Dantas

adantas@ctvc.pt

ProDEI, Universidade do Porto
Portugal

Carlos Baquero

cbm@fe.up.pt

Universidade do Porto & INESC TEC
Portugal

Abstract

Virtual presence demands ultra-low latency, a factor that centralized architectures, by their nature, cannot minimize. Local peer-to-peer architectures offer a compelling alternative, but also pose unique challenges in terms of network infrastructure.

This paper introduces a prototype leveraging Conflict-Free Replicated Data Types (CRDTs) to enable real-time collaboration in a shared virtual environment. Using this prototype, we investigate latency, synchronization, and the challenges of decentralized coordination in dynamic non-Byzantine contexts.

We aim to question prevailing assumptions about decentralized architectures and explore the practical potential of P2P in advancing virtual presence. This work challenges the constraints of mediated networks and highlights the potential of decentralized architectures to redefine collaboration and interaction in digital spaces.

Keywords—Virtual Reality, Peer-to-Peer, Conflict-Free Replicated Data Types, Low Latency, Collaborative Systems

1 Introduction

Latency remains a fundamental obstacle in achieving immersion in VR. Existing cloud-based architectures introduce network delays that can disrupt user experience and exacerbate VR sickness. On the other hand, local peer-to-peer (P2P) collaboration promises ultra-low latency, but its adoption has been stifled by challenges like NAT traversal and limited support in public and semi-public networks. To the best of our knowledge, this paper represents the first exploration of Conflict-Free Replicated Data Types (CRDTs) within the context of VR.

We hypothesize that integrating P2P architectures with Conflict-Free Replicated Data Types will minimize latency in these environments. This paper evaluates the impact of CRDT integration on user experience and its broader implications for the development of real-time, dynamic collaborative environments. This leads us to the following research questions:

- Can CRDTs serve as a foundational technology for collaboration in P2P VR systems?
- What is the magnitude of network latency reduction achievable with P2P architectures?

We will thus focus on the technical evaluation of P2P architectures and CRDTs for VR collaboration; however, the broader implications of architectural paradigms applied to VR, particularly the contrast between centralized and decentralized systems, are hard to ignore. As virtual environments become central to work and social interactions, centralized networks, such as the Meta VR ecosystem, pose risks of restricted access and data exploitation [11]. Decentralized architectures are essential to safeguard equitable and open digital spaces. This paper explores the implementation and performance of these technologies, offering a foundation to address challenges and inform the design of future VR systems.

2 Background and Related Work

2.1 Network Latency in VR

Network latency is critical for collaborative VR systems, particularly with increasing synchronized data volumes. Photon-to-motion latency exceeding 100 ms significantly degrades user experience [5, 8, 14]. Similarly, network delays over 230 ms impair task performance [7] and disrupt shared presence in multiuser VR [20].

P2P networks can reduce latency and cost by leveraging user devices instead of centralized servers. In MMOGs (massively multiplayer online games), hybrid P2P-edge server approaches reduce latency by up to 45% [17]. While P2P transmission strategies address bandwidth bottlenecks [10], P2P systems face challenges with NAT traversal and firewalls [21].

2.2 Conflict-Free Replicated Data Types (CRDTs)

CRDTs ensure eventual consistency through independent updates and conflict resolution without centralized coordination [3]. They are categorized as state-based, with high communication overhead, and operation-based, which are bandwidth-efficient but depend on reliable delivery [19]. Delta State CRDTs (δ -CRDTs) combine these strengths by transmitting compact delta-states [4].

CRDTs are widely used in distributed systems, including Riak, Redis CRDBs, Azure Cosmos, and collaborative tools like Yjs and Automerger [12]. Automerger, while capable of operating in P2P contexts, reflects skepticism about the practicality of pure P2P systems in consumer applications, favoring architecture-agnostic designs [21]. Similarly, Yjs

excels in browser-based environments, but has not been optimized for dynamic, game-like VR settings, highlighting a gap for further exploration.

3 Methodology

We employed iterative prototype development to investigate latency and synchronization in VR collaboration. By **reframing latency as a data consistency challenge**, we explored CRDTs as a solution for decentralized collaboration. Each iteration addressed practical issues, such as peer-to-peer synchronization and conflict resolution, while evaluating their influence on real-time performance and the applicability of CRDTs in dynamic environments.

In this initial exploration, we focused primarily on the latency and effectiveness of CRDTs in a real-time virtual reality environment. Our approach proved to be robust against temporary network partitions. However, aspects such as synchrony conditions were not addressed within the scope of this study. The real-time nature of the application and the continuous broadcast of updates effectively mitigated many network-related challenges, resulting in a seamless user experience. Investigating the system's behavior under more severe conditions, such as prolonged network partitions or significant message loss, could offer deeper insights into how consistency and reliability manifest in this type of architecture.

4 BrickSync: VR P2P CRDTs in Action

BrickSync is an initial exploration of a framework for CRDT-based game-state synchronization, implemented within the Unity game engine, which allows us to collaborate in VR without a server.

Users interact with the environment using standard Meta Quest 3 controls, with support for both hand tracking and controller input. This allows users to grab and manipulate bricks using one or both hands.

Users collaboratively manipulate bricks in a shared environment, and their interactions are synchronized through CRDTs. The virtual scene consists of a table, a button for spawning bricks, and the ability to collaboratively build structures by moving and aligning bricks.

To assist in connection monitoring, a fixed position video feed window hovers near the main interaction area. This real-time feed displays what other users are seeing. In VR, where the headset obscures the view-port, this feature helps participants understand the actions of others and effectively assess the connection status.

Additional debugging and control tools include: real-time ping monitoring, an average ping record over the last minute, a connection status display for all WebRTC data channels, and a window for viewing local logs.

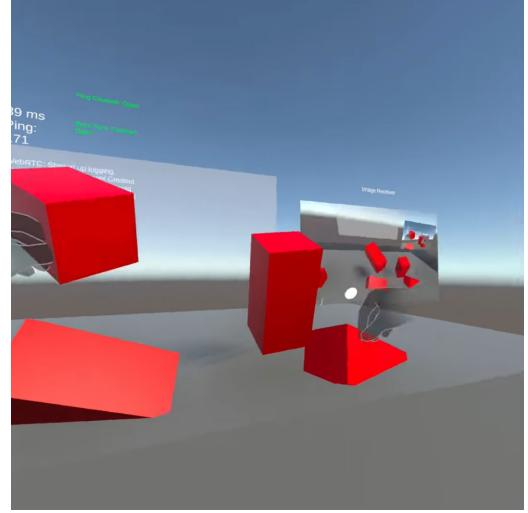


Figure 1. A snapshot of BrickSync in action, showcasing two users collaboratively manipulating virtual objects in a shared VR environment. Real-time synchronization of object states is achieved using CRDTs over a P2P WebRTC connection.

4.1 Communication and Networking

In BrickSync, the communication protocol has 2 elements:

- 1. Peer Discovery:** A WebSocket signaling server is used to establish connections between peers. Once connected, the signaling server is no longer involved in data exchange between peers.
- 2. P2P Channels:** WebRTC data channels are used for real-time communication between peers. These channels transmit CRDT updates, enabling synchronized state management across all connected devices.

4.1.1 Peer Discovery. The peer discovery process begins with one peer creating an SDP (Session Description Protocol) offer and sending it to another peer via a signaling server. The receiving peer responds with its own SDP, and both exchange ICE (Interactive Connectivity Establishment) candidates to establish a direct WebRTC connection. Once the connection is established, the signaling server is no longer involved, serving only to facilitate the initial handshake and maintain awareness of peer activity.

We recognize that using a signaling server deviates from a pure peer-to-peer (P2P) setup. This choice was driven by time and scope constraints, as implementing fully decentralized P2P signaling was beyond the objectives we set out to achieve. The challenge of bootstrapping nodes in a peer-to-peer network has long been recognized. For example, Kademlia assumes that at least one known node is needed for initialization [15].

In general, research has shown that the bootstrap problem often necessitates trade-offs between decentralization,

scalability, and complexity. For example, leveraging auxiliary systems like public overlays provides practical solutions to bootstrap peers under constraints such as NAT traversal, but often requires additional infrastructure and operational overhead [22]. Other approaches, such as employing distributed mechanisms like DNS or DHT entries, rely on peers maintaining accurate and up-to-date information about the network [13]. While alternative methods like Bluetooth or blockchain-based mempool signaling could theoretically eliminate server reliance, they would fundamentally shift the focus and complexity of this work.

Instead, we opted for a lightweight signaling server, deployed locally and remotely on Azure at zero cost. The server's sole responsibility is to facilitate peer discovery, without influencing or participating in the subsequent data exchange.

4.1.2 P2P Communication. Our WebRTC data channels rely on SCTP (Stream Control Transmission Protocol) and support both ordered and unordered delivery. For CRDT synchronization, we configured the channel for ordered delivery, achieving source FIFO (First In, First Out) order.

Previous research has explored WebRTC for peer-to-peer collaborative editing in browsers [16], highlighting its potential despite early limitations on mobile devices. However, since then WebRTC mobile support has expanded, making it a viable option for real-time communication on smartphones. This is particularly relevant for VR applications on devices like the Meta Quest 3, which runs on Android and supports WebRTC. Thus, our choice of WebRTC aligns with the growing capabilities of mobile VR platforms.

The local RTC peer connections were also initialized with the option to use a STUN server, such as Google's public STUN servers. This facilitates the discovery of public IP addresses and port mappings, resolving NAT traversal issues in site-to-site communication. This is particularly useful in semi-public networks, such as academic environments like Eduroam [2], where peers are often behind NATs that block direct connections [18]. Although not strictly needed in controlled environments, the use of a STUN server is a valuable tool for testing in heterogeneous network configurations.

Each peer is also configured to handle incoming messages from other peers. This includes rendering the video channel to the video feed window and relaying data channel messages to the CRDT logic.

To optimize communication, we implemented separate data channels: one for Round-Trip Time (RTT) measurements and another for CRDT updates. The RTT channel tracks latency, while the CRDT channel is used for the synchronization of the in-game bricks.

4.2 CRDT Integration

Our CRDT implementation evolved through multiple phases. We first explored a naive operation-based approach and

later a state-based approach. In our implementation, a Unity MonoBehaviour operates as adapter [9] between the representation in the game engine and the CRDT. A local map per replica is also used to keep track of all the bricks – and the corresponding CRDTs – as they are spawned.

4.2.1 Operation based approach. We initially chose an operation-based approach with the goal of reducing communication overhead while maintaining consistent, frame-by-frame updates. In this approach, each update is represented as an *operation* broadcast to all replicas, which then apply the operations. To accommodate users' real-time, continuous interactions with bricks, translation offsets are calculated and transformed into discrete operations.

In order to ensure that each operation is transmitted maintaining causal order, we used Vector Clocks to track the order of operations, allowing us to determine whether an incoming operation has already been applied, is not yet ready to be applied (due to unmet dependencies), or is the next in the logical sequence. In the latter case, the operation can be applied to the local replica.

Although operation-based CRDTs can perform well for real-time interactions over reliable channels, in practice, reliable channels are hard to come by and can be a source of extra latency. While the usage of vector clocks together with the FIFO ordering afforded by WebRTC proved sufficient for a two-user interaction, to achieve reliability beyond point-to-point or small groups, a group membership mechanism is necessary to properly handle view changes. Tools like Spread [1] can be used to provide reliable causal ordering and possibly overcome the need for vector clocks as ordering would be relegated to the middleware.

Causal delivery, either via specialized middleware or by application level ordering with vector clocks, might deliver concurrent operations in different order at different replicas. This requires special handling at the application layer unless these operations are already commutative.

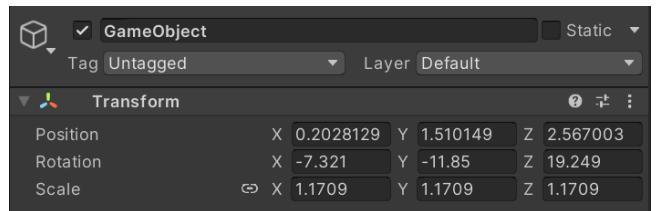


Figure 2. A Unity GameObject Transform component. Rotation is non-commutative: the order of rotations affects the final orientation. Rotation is a quaternion (4D space to prevent gimbal lock) represented in Euler angles on the UI.

For example, in BrickSync, non-commutative operations – such as rotating an object as highlighted in Figure 2, can lead to divergent outcomes if applied in different orders. Within

operation-based CRDTs, this issue can be tackled using one of two strategies:

- Canonical Ordering: For example a Last-Writer-Wins (LWW) approach, which would avoid surprise merges by making one rotation “win” for concurrent actions. UX is not truly collaborative.
- Specialized Handling: For example averaging the rotations or accumulating deltas. This is more collaborative, but, might feel non intuitive if the final orientation is unexpected.

These approaches to handling non-commutative operations are inherently limited, and expose another drawback of operation-based CRDTs: without a comprehensive chronicle of the object, we are constrained on the strategies for reconciling conflicts. Although operation-based CRDTs still operate on a locally maintained state, and this local state could theoretically support techniques like representing divergent views using metadata, doing so would complicate the model and blur the boundaries of what defines an operation-based CRDT [6]. State-based CRDTs, on the other hand, propagate the entire state and, in this case, we found that they provide a better framework for ensuring an adequate handling of commutativity.

As we will see next, state-based CRDTs allow replicas to reconcile divergent states by encoding discrepancies in auxiliary properties – additional metadata or dimensions separate from the conflicting attributes. In the context of real-time dynamic collaboration, this approach—at the expense of increased bandwidth and storage requirements—supports a broader range of conflict resolution strategies, including prioritization, blending, or heuristic-driven reconciliation.

4.2.2 State based approach. Faced with the aforementioned limitations of operation-based CRDTs – (1) reliance on causal order and commutative operations and (2) the lack of a full chronicle of the object (the object’s complete state history, including all changes and associated metadata) – we opted to implement a state-based approach. Based off Unity’s native data structures, we developed a MV-Transformer (Figure 3), a higher-level state-based CRDT designed to encapsulate and synchronize a GameObject’s Transform state. In addition to mediating state resolution for position, rotation, and scale, the MV-Transformer incorporates a register for tracking which replica(s) are currently manipulating the object – in other words, which users are holding on to the object using the Meta Quest 3 controls. The MV-Transformer was engineered to toggle between two synchronization strategies for the underlying vectors: local-space updates and world-space updates. The local-space strategy, inspired by PN-Counters – a type of CRDT that track increments and decrements separately, allowing for consistent concurrent updates to numerical values – applies synchronization based on offset-based updates relative to the object’s last known position.

In contrast, the world-space strategy, modeled after a Last-Writer-Wins (LWW) approach, synchronizes using the latest absolute position of the object in world space. Instead of relying on a traditional feature flag to toggle between these strategies, the MV-Transformer also contains an internal boolean register that can dynamically dictate the synchronization mode (local-space or world-space).

To clarify the difference between the synchronization strategies, consider **world-space mode**. When multiple users interact with the same object in world-space mode, the object appears to oscillate between their control. Once one of the users releases their grip, the object returns to the hand of the other user — the one who hold on to the object the longest. This simple method of conflict resolution is effective in scenarios where users are collaborating and actively communicating, as they can quickly recognize the conflict and coordinate who should release the object.

In **local mode**, on the other hand, synchronization relies on offset-based updates relative to the object’s last known position. This approach prevents oscillations but may require more sophisticated handling of positional drift between replicas.

Through this prototype, we found that the effectiveness of CRDT-based architectures hinges on how divergences – such as conflicting updates or superimposed states – are presented to users. Providing clear, consistent, and navigable representations of these inconsistencies is crucial for collaboration.

5 Findings and Discussion

This section discusses key findings and challenges from the prototype development, focusing on technical and UX observations from the development process (A), conflict resolution in collaborative interactions in VR (B) and performance and latency analysis (C).

5.1 Technical and UX Observations

The development process revealed several technical and UX challenges.

5.1.1 CRDT Developer Experience. The elegance of CRDTs made implementing the shared state intuitive. Synchronization occurred naturally, creating the illusion of a shared, independent space. Testers often expressed surprise at the absence of a centralized server, and that you can conduct the experience with just two VR headsets – even without an internet connection. We interpret this as a positive indication that CRDTs are a natural fit for architecting shared game-state in collaborative systems. All other technical challenges relate to establishing and maintaining the P2P connections.

5.1.2 Heterogeneous Network Behavior. Different network environments yielded varying behaviors. Testing in

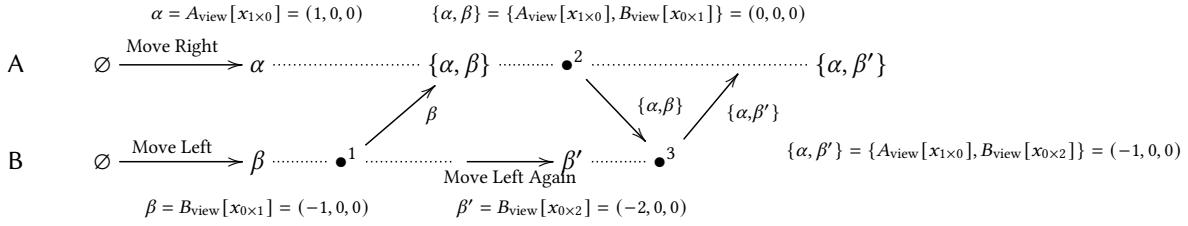


Figure 3. MV-Transformer synchronization in Local-Space Mode with replicas A and B. A applies 'Move Right' to reach $(1,0,0)$; B applies 'Move Left' twice to $(-2,0,0)$. Solid arrows show local updates, dotted lines indicate state exchanges. The process merges to a final state of $(-1,0,0)$.

the semi-public networks caused erratic system behavior, creating confusion. This was mitigated with:

- Resorting to a mobile hotspot for a stable testing environment.
- Hosting a remote WebSocket server on Azure for consistent signaling.
- Using a STUN server to address NAT traversal issues.

This experience highlights pain points in P2P systems when deployed to some networks and underscores the need for controlled network configurations to ensure consistent performance.

5.1.3 Closing Data Channels. WebRTC data channels closed arbitrarily on Meta Quest 3 devices, whereas this behavior was not observed during PC testing. The issue was mitigated by implementing a renegotiation strategy to reopen channels when they closed unexpectedly. While effective, this approach introduced additional latency and required specifying channel IDs during initialization. We speculate that the behavior stems from Android security policies or specific constraints within Meta Quest devices.

Such erratic channel closures and the resulting variability in delivery order have implications for idempotency and retransmission decisions. To address these challenges, state-based CRDTs offer robustness by ensuring consistency even under unpredictable network conditions or device-specific issues like those observed with Meta Quest 3.

Regarding fault tolerance, our system leverages the inherent properties of CRDTs to handle network partitions. Replicas can operate independently and merge their states upon reconnection, ensuring eventual consistency. For scenarios with high packet loss, the state-based CRDT approach provides robustness by allowing full state retransmission, while the ordered delivery of WebRTC helps maintain consistency for operation-based CRDTs. However, we assume a non-Byzantine environment, meaning malicious peers are not accounted for in the current design. Addressing such adversarial scenarios would require additional security mechanisms, which are beyond the scope of this initial exploration.

While CRDTs provide a solid foundation for fault tolerance, extreme edge cases necessitate additional research.

5.2 Global Rules

In this section, we introduce the term *Global Rules* to describe any state transformation that is not directly authored by a specific user. These transformations can be continuous – such as gravity, which requires real-time updates – or discrete – such as a wind direction change or the spawning of an enemy in a game. They may also involve complex causal chains. For example, a user releases an object under the influence of gravity, which collides with a stack of dominoes initiating a chain reaction.

We refer to these as *Global Rules* rather than environmental or physics rules, as they encompass any game logic capable of altering the game state without a specific author. While this touches on broader challenges in non-server-authoritative game engine design rather than solely state synchronization with CRDTs, we believe this perspective can inform the application of CRDTs in this context.

5.3 Conflict Visualization

When two users manipulated the same object simultaneously under world-space mode, the object oscillated between positions as the CRDT logic attempted to reconcile superimposed states. While this behavior did not significantly affect the user experience (on the contrary), it did raise questions about conflict resolution in collaborative contexts in general.

In the case of BrickSync we identified several strategies to address the issue with oscillation:

- Prioritizing the most recent update (last-writer-wins).
- Introducing heuristics to maximize collaborative convergence, such as favoring positions that align with shared goals like building a straight wall.
- Averaging positions. In this case if two users manipulate the same object in different directions it should stand in the middle.
- Applying constraints to prevent simultaneous manipulations – this one is a last resort.

Building on this, we propose embedding Dynamic Strategy Switching directly within CRDTs. This involves designing the CRDT to adaptively transition between conflict resolution strategies based on contextual factors such as interaction patterns, user intent, or system state. For example:

The CRDT could default to heuristic-based reconciliation, or last-writer-wins during cooperative tasks but switch to constraint-based mechanisms in competitive or high-conflict scenarios. Thresholds or triggers, such as the frequency of conflicting updates or task-specific goals, would determine when and how the strategy changes.

Other heuristic-based conflict resolution strategies could employ a prediction layer, for example applying dead reckoning principles – a method commonly used in fast-paced multiplayer games to estimate future positions of moving objects. By integrating this layer into the CRDT merge process, states aligning with the predicted path of an object could be prioritized to promote a smoother experience.

Heuristic-based approaches are directly applicable to existing frameworks such as Automerger or Yjs. For example, in a collaborative editing scenario, if two users are making conflicting contributions to the same text, a semantic heuristic could be applied to arbitrate, prioritizing the changes that align more closely with the document's intent or structure. Alternatively, edits could be merged contextually based on their relevance to the overall goal, such as prioritizing keywords or stylistic consistency. This will become increasingly relevant as collaborative editing systems evolve to include real-time AI agents, where clear feedback and effective merging strategies for individual contributions are key.

5.4 Performance and Latency Analysis

As part of our experiment with BrickSync, we collected latency data. Network latency was measured across various configurations, as shown in Table 5 and illustrated in Figure 4 and 6.

While delays were noticeable in the WebRTC video stream, brick interactions remained fluid due to:

1. Real-time CRDT updates providing immediate feedback for local actions.
2. Users' inability to distinguish between network delay and the natural delay of other users' actions.

By prioritizing local updates, CRDTs **minimize perceptible latency and mitigate VR sickness** caused by visual disconnects between user actions and system feedback.

While our experiments demonstrate the feasibility of CRDT-based synchronization in a two-user scenario, we acknowledge that this does not reflect the scalability required for real-world applications. The current setup was constrained by resources and aimed at initial exploration. Moreover, the choice of CRDT type has implications for scalability. State-based CRDTs, while simpler to implement, can lead to increased bandwidth usage due to the transmission of full

states. Operation-based CRDTs, on the other hand, require reliable broadcast mechanisms, which can be complex to manage. Additionally, the use of vector clocks for maintaining causal order can result in significant memory and message size overhead as the number of replicas grow. These challenges were not pronounced in our two-user setup but need to be addressed, potentially through the adoption of delta-based CRDTs or other optimized approaches.

Connection Type	WebRTC P2P	Local WebSocket	Azure-Hosted WebSocket
PC to PC - Ethernet	18 ms	45 ms	155 ms
PC to PC - WiFi	25.75 ms	71.5 ms	174 ms
Quest to PC - WiFi	46 ms	50.5 ms	165.5 ms
Quest to Quest - WiFi	87.5 ms	111.75 ms	220.5 ms
Quest to Quest - Hotspot	74 ms	215 ms	236.25 ms

Figure 5. Average latency measurements for different connection types and configurations across all test scenarios.

It's also worthwhile to note that in scenarios with high number of replicas vector clocks become inefficient due to their size, that scales linearly with the number of nodes, resulting in significant metadata and communication overhead. Delta-state CRDTs can reduce bandwidth by transmitting only incremental updates, but still require end-to-end mechanisms to guarantee causal consistency.

Furthermore, we observe that empirical user studies should be conducted to validate the collaborative experience, perceived latency, and conflict resolution mechanisms. Understanding user feedback, particularly regarding phenomena like object oscillation in world-space mode, is crucial for refining the system. Our priority in this initial exploration was to establish technical feasibility and user feedback was sought but collected in a limited way.

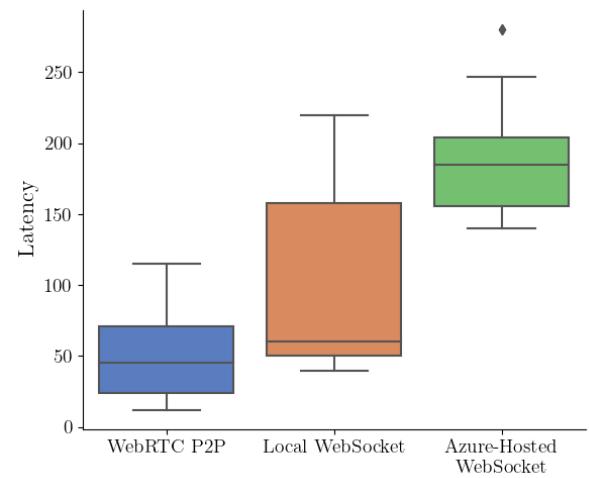


Figure 6. Distribution of latency results based on underlying architecture.

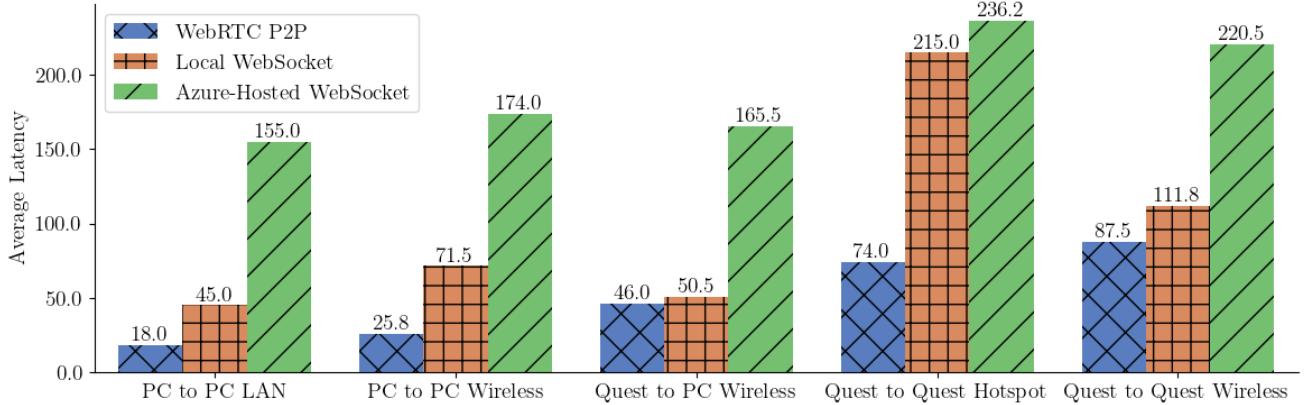


Figure 4. Impact of Connection Type and Architecture on Latency. The x axis displays different network configurations groups that show the different architectures, as expected, P2P is much faster.

In addition to these measurements, we conducted further tests to evaluate the impact of payload size. Unfortunately, WebRTC imposes a maximum payload size of 16 KB. And although comparative results show promising trends, to fully understand the implications, it would be necessary to implement message chunking to accommodate payloads exceeding the 16 KB limit, and conduct more tests to reduce the impact of outliers and network conditions.

6 Conclusions

This study demonstrates the feasibility of using CRDTs for decentralized VR collaboration within a P2P architecture. Our prototype demonstrates the ability to achieve low-latency, synchronized interactions without reliance on centralized servers. The controlled co-located VR environment served as a practical testbed to consider latency in a CRDT-based architecture.

Our findings demonstrate a substantial reduction in network latency when using P2P architectures compared to remote server setups. Specifically, the average latency in P2P configurations is approximately 50 ms, a 75% reduction from the 200 ms observed in remote server connections. Moreover, P2P achieved the lowest recorded latency at 18 ms, compared to 45 ms on WebSocket-based setups, showcasing its ability to outperform other architectures by a factor of 2 in optimal conditions. While the use of more performant and well-connected remote servers may mitigate these differences, such configurations often incur higher costs and dependency on centralized infrastructure. P2P emerges as the most cost-effective solution for reducing latency, aligning the low-latency requirements of immersive VR environments with decentralization.

The interaction between CRDTs and Global Rules, such as gravity, revealed opportunities in handling non-user-authored

state, revealing the importance of integrating adaptive conflict resolution strategies, such as dynamic strategy switching, directly into CRDTs. The findings suggest that decentralized systems must address these dynamics to support emergent behaviors in collaborative environments.

We further identified key challenges and opportunities in extending CRDTs and P2P systems to game-like, dynamic environments, particularly in conflict arbitration and scalability. We invite researchers and practitioners to build on these findings, exploring broader applications of decentralized architectures in VR and other collaborative settings. The potential of these systems to redefine collaboration and interaction in digital spaces is immense, and their future development holds promise for transformative impacts across industries and scientific domains.

7 Future Work

Our exploratory work revealed several promising directions to advance decentralized realtime collaborative environments, including local-first principles applied to VR and proximity-aware network topologies for in-loco coordination. However, the most critical avenue lies in embedding dynamic strategies, such as switching between heuristics and constraints based on interaction context, as discussed in Section V. Specifically, we hope to continue exploring how local-first CRDT designs can handle non-user-authored changes, such as physics-based interactions or other global rules.

Scalability remains a critical area for further research. Experiments with a larger number of users and objects will be essential to understand the system's performance in more complex VR environments.

As decentralized architectures challenge centralized models, these efforts could pave the way for new interaction paradigms, positioning CRDTs as foundational building blocks for the open digital commons.

8 Acknowledgments

This work was partially financed by National Funds through the Portuguese funding agency, FCT (Fundação para a Ciência e a Tecnologia) under project LA/P/0063/2020 – DOI 10.54499/LA/P/0063/2020¹. We also thank the PRODEI Doctoral Program (Programa Doutoral em Engenharia Informática) at the Faculty of Engineering, University of Porto, and CTVC (Cooperativa Tecnológica de Viana do Castelo, a tech cooperative in northern Portugal) for their support and collaboration.

References

- [1] [n. d.]. The Spread Toolkit. <https://www.spread.org/>. Accessed: January 9, 2025.
- [2] 2025. eduroam - Education Roaming. <https://eduroam.org/>. Accessed: 2025-01-09.
- [3] Paulo Sérgio Almeida. 2024. Approaches to Conflict-free Replicated Data Types. *Comput. Surveys* (Sept. 2024), 3695249. <https://doi.org/10.1145/3695249>
- [4] Paulo Sérgio Almeida, Ali Shoker, and Carlos Baquero. 2018. Delta State Replicated Data Types. *J. Parallel and Distrib. Comput.* 111 (Jan. 2018), 162–173. <https://doi.org/10.1016/j.jpdc.2017.08.003>
- [5] Muhammad Danish Affan Anua, Ismahafezi Ismail, Nur Saadah Mohd Shapri, Maizan Mat Amin, and Mohd Azhar M. Arsal. 2022. A Systematic Review of Purpose and Latency Effect in the Virtual Reality Environment. In *Intelligent Technologies for Interactive Entertainment*, Zhihan Lv and Houbing Song (Eds.). Vol. 429. Springer International Publishing, Cham, 403–413. https://doi.org/10.1007/978-3-030-99188-3_25
- [6] Carlos Baquero, Paulo Sérgio Almeida, and Ali Shoker. 2014. Making Operation-Based CRDTs Operation-Based. In *Distributed Applications and Interoperable Systems*, Kostas Magoutis and Peter Pietzuch (Eds.). Vol. 8460. Springer Berlin Heidelberg, Berlin, Heidelberg, 126–140. https://doi.org/10.1007/978-3-662-43352-2_11
- [7] Armin Becher, Jens Angerer, and Thomas Grausopf. 2020. Negative Effects of Network Latencies in Immersive Collaborative Virtual Environments. *Virtual Reality* 24, 3 (Sept. 2020), 369–383. <https://doi.org/10.1007/s10055-019-00395-9>
- [8] Polona Caserman, Michelle Martinussen, and Stefan Göbel. 2019. Effects of End-to-end Latency on User Experience and Performance in Immersive Virtual Reality Applications. In *Entertainment Computing and Serious Games*, Erik Van Der Spek, Stefan Göbel, Ellen Yi-Luen Do, Esteban Clua, and Jannicke Baalsrud Hauge (Eds.). Vol. 11863. Springer International Publishing, Cham, 57–69. https://doi.org/10.1007/978-3-030-34644-7_5
- [9] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1994. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.
- [10] Yonghao Hu, Zhaojun Chen, Xiaojun Liu, Fei Huang, and Jinyuan Jia. 2017. WebTorrent Based Fine-Grained P2P Transmission of Large-Scale WebVR Indoor Scenes. In *Proceedings of the 22nd International Conference on 3D Web Technology*. ACM, Brisbane Queensland Australia, 1–8. <https://doi.org/10.1145/3055624.3075944>
- [11] Khari Johnson. 2024. Meta's VR Headset Harvests Personal Data Right Off Your Face. *Wired* (2024). <https://www.wired.com/story/metas-vr-headset-quest-pro-personal-data-face/> Accessed: 2025-01-13.
- [12] Martin Kleppmann, Adam Wiggins, Peter Van Hardenberg, and Mark McGranaghan. 2019. Local-First Software: You Own Your Data, in Spite of the Cloud. In *Proceedings of the 2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*. ACM, Athens Greece, 154–178. <https://doi.org/10.1145/3359591.3359737>
- [13] Mirko Knoll, Arno Wacker, Gregor Schiele, and Torben Weis. 2008. Bootstrapping in Peer-to-Peer Systems. In *2008 14th IEEE International Conference on Parallel and Distributed Systems*. IEEE, Melbourne, Australia, 271–278. <https://doi.org/10.1109/ICPADS.2008.26>
- [14] Marc Erich Latoschik, Florian Kern, Jan-Philipp Stauffert, Andrea Bartl, Mario Botsch, and Jean-Luc Lugrin. 2019. Not Alone Here?! Scalability and User Experience of Embodied Ambient Crowds in Distributed Social Virtual Reality. *IEEE Transactions on Visualization and Computer Graphics* 25, 5 (May 2019), 2134–2144. <https://doi.org/10.1109/TVCG.2019.2899250>
- [15] Petar Maymounkov and David Mazières. 2002. Kademia: A Peer-to-Peer Information System Based on the XOR Metric. In *Peer-to-Peer Systems*, Gerhard Goos, Juris Hartmanis, Jan Van Leeuwen, Peter

¹<https://doi.org/10.54499/LA/P/0063/2020>

- Druschel, Frans Kaashoek, and Antony Rowstron (Eds.). Vol. 2429. Springer Berlin Heidelberg, Berlin, Heidelberg, 53–65. https://doi.org/10.1007/3-540-45748-8_5
- [16] Brice Nédelec, Pascal Molli, and Achour Mostéfaoui. 2016. CRATE: Writing Stories Together with Our Browsers. *WWW (Companion Volume)* (2016), 231–234.
- [17] Jared N. Plumb, Sneha Kumar Kasera, and Ryan Stutsman. 2018. Hybrid Network Clusters Using Common Gameplay for Massively Multiplayer Online Games. In *Proceedings of the 13th International Conference on the Foundations of Digital Games*. ACM, Malmö Sweden, 1–10. <https://doi.org/10.1145/3235765.3235785>
- [18] Jonathan Rosenberg, Rohan Mahy, Philip Matthews, and David Wing. 2008. RFC 5389: Session Traversal Utilities for NAT (STUN). <https://www.rfc-editor.org/rfc/rfc5389>. Accessed: 2025-01-08.
- [19] Marc Shapiro, Nuno Preguiça, Carlos Baquero, and Marek Zawirski. 2011. Conflict-Free Replicated Data Types. In *Stabilization, Safety, and Security of Distributed Systems*, Xavier Défago, Franck Petit, and Vincent Villain (Eds.). Vol. 6976. Springer Berlin Heidelberg, Berlin, Heidelberg, 386–400. https://doi.org/10.1007/978-3-642-24550-3_29
- [20] Mel Slater, Beau Lotto, Maria Marta Arnold, and Maria V. Sanchez-Vives. 2009. How We Experience Immersive Virtual Environments: The Concept of Presence and Its Measurement. *Anuario de psicología / The UB Journal of psychology* 40, 2 (Dec. 2009), 193–210.
- [21] Peter Van Hardenberg and Martin Kleppmann. 2020. PushPin: Towards Production-Quality Peer-to-Peer Collaboration. In *Proceedings of the 7th Workshop on Principles and Practice of Consistency for Distributed Data*. ACM, Heraklion Greece, 1–10. <https://doi.org/10.1145/3380787.3393683>
- [22] D. I. Wolinsky, P. St. Juste, P. O. Boykin, and R. Figueiredo. 2010. Addressing the P2P Bootstrap Problem for Small Overlay Networks. In *2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P)*. IEEE, Delft, Netherlands, 1–10. <https://doi.org/10.1109/P2P.2010.5569960>

A Additional Data and Figures

Below are additional figures and detailed latency measurements across all test scenarios. As well as some implementation and testing details.

Date	Scenario	Connection Type	WebRTC P2P	Local WebSocket	Azure-Hosted WebSocket
10-01-2025	Minimal Payload	PC to PC LAN	20 ms	45 ms	180 ms
10-01-2025	Minimal Payload	PC to PC Wireless	25 ms	60 ms	140 ms
10-01-2025	Minimal Payload	Quest to PC Wireless	50 ms	47 ms	150 ms
10-01-2025	Minimal Payload	Quest to Quest Wireless	70 ms	180 ms	220 ms
10-01-2025	Minimal Payload	Quest to Quest Hotspot	75 ms	200 ms	280 ms
14-12-2024	Minimal Payload	PC to PC LAN	12 ms	45 ms	140 ms
14-12-2024	Minimal Payload	PC to PC Wireless	22 ms	101 ms	158 ms
14-12-2024	Minimal Payload	Quest to PC Wireless	34 ms	55 ms	177 ms
14-12-2024	Minimal Payload	Quest to Quest Wireless	55 ms	150 ms	247 ms
14-12-2024	Minimal Payload	Quest to Quest Hotspot	41 ms	220 ms	280 ms
08-01-2025	16kb Payload	PC to PC LAN	20 ms	40 ms	150 ms
08-01-2025	16kb Payload	PC to PC Wireless	30 ms	60 ms	200 ms
08-01-2025	16kb Payload	Quest to PC Wireless	50 ms	50 ms	170 ms
08-01-2025	16kb Payload	Quest to Quest Wireless	110 ms	57 ms	215 ms
08-01-2025	16kb Payload	Quest to Quest Hotspot	90 ms	220 ms	190 ms
14-12-2024	16kb Payload	PC to PC LAN	20 ms	50 ms	150 ms
14-12-2024	16kb Payload	PC to PC Wireless	26 ms	65 ms	198 ms
14-12-2024	16kb Payload	Quest to PC Wireless	50 ms	50 ms	165 ms
14-12-2024	16kb Payload	Quest to Quest Wireless	115 ms	60 ms	200 ms
14-12-2024	16kb Payload	Quest to Quest Hotspot	90 ms	220 ms	195 ms

Figure 7. Average latency measurements for different connection types, scenarios, and configurations across all test scenarios.

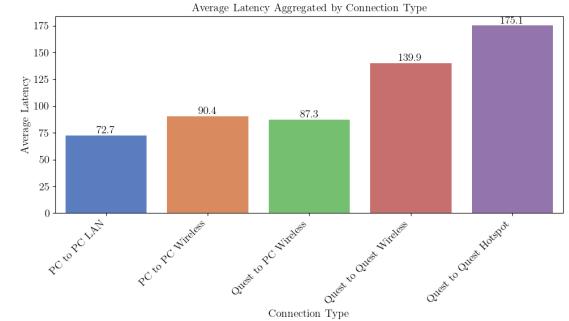


Figure 8. Average Latency Across Connection Types: PC to PC LAN shows the lowest latency, while Quest to Quest Hotspot exhibits the highest.

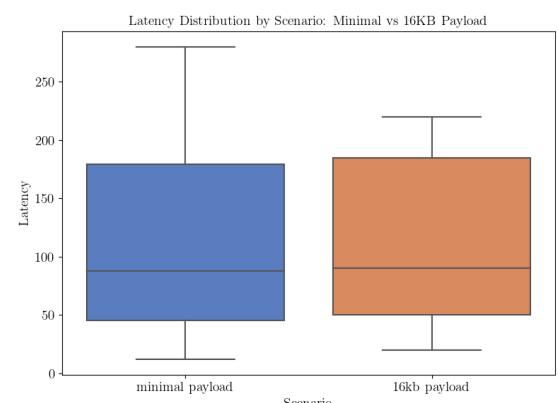


Figure 9. Latency Distribution Across Scenarios with Outliers: Minimal payload shows slightly lower median latency compared to 16KB payload. A larger sample is necessary to understand trends, as WebSocket's message prioritization might influence results.

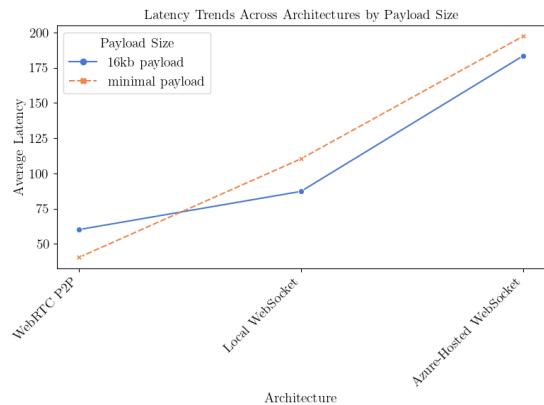


Figure 10. Latency Trends by Architecture and Payload Size:
Minimal payload is only faster on WebRTC P2P, while 16KB payload performs better on WebSocket architectures.

Networks

1. Commercial office network (Vodafone 5GHz, 500Mbps)
2. Mobile hotspot (NOS 5G, Xiaomi Mi 11T)

Devices

1. PC (Windows 11, Ryzen 5 5500, 32GB RAM)
2. MacBook Air (macOS Sonoma, M1, 8GB RAM)
3. Meta Quest 3 (Android 11, Snapdragon XR2, 8GB RAM)

Table 1. Listing of networks and devices used in the experiment.

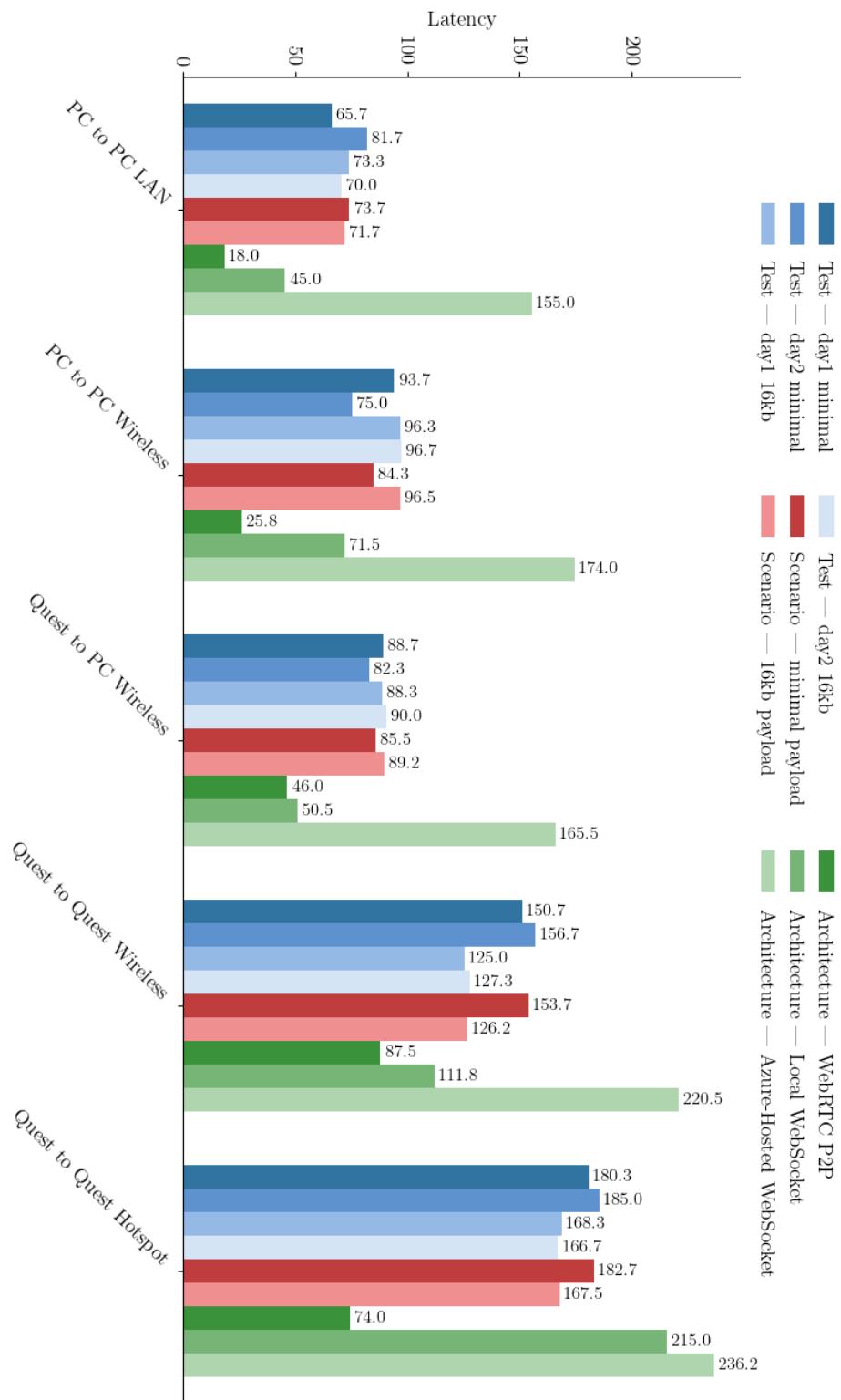


Figure 11. Breakdown across connection types, test scenarios, payload sizes, and architectures.