UNIVERSIDAD DE GUANAJUATO

DIVISIÓN DE INGENIERÍAS
CAMPUS IRAPUATO-SALAMANCA

DISEÑO Y EVALUACIÓN DE UN MODELO DE COLABORACIÓN MULTIMODAL, HETEROGÉNEO Y DISTRIBUIDO PARA ENTORNOS INMERSIVOS

TESIS

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN INGENIERÍA ELÉCTRICA

OPCIÓN: INSTRUMENTACIÓN Y SISTEMAS DIGITALES

PRESENTA:

ING. GUSTAVO ADOLFO MURILLO GUTIERREZ

ASESOR:

DR. JUAN PABLO IGNACIO RAMÍREZ PAREDES

CO-ASESOR:

DR. URIEL HAILE HERNÁNDEZ BELMONTE

SALAMANCA, GTO. ENERO, 2026

Agradecimientos institucionales

A la Secretaría de Ciencia, Humanidades, Tecnología e Innovación (Secihti), por el invaluable apoyo económico mediante la beca con registro ligado al No. CVU 1320374 (2023-2025), que hizo posible mi formación en el programa de Maestría en Ingeniería Eléctrica. Este trabajo representa el compromiso adquirido con su respaldo institucional.

Ciencia y Tecnología
Secretaría de Ciencia, Humanidades, Tecnología e Innovación

A la Universidad de Guanajuato, particularmente a la División de Ingenierías del Campus Irapuato–Salamanca, por haberme abierto las puertas de su posgrado y brindarme una formación académica en el campo de la Instrumentación y Sistemas Digitales, que hoy culmina con la presentación de esta tesis.

UNIVERSIDAD DE GUANAJUATO

Al Laboratorio de Visión, Robótica e Inteligencia Artificial (LaViRIA), por proporcionarme acceso a sus instalaciones y equipos especializados, recursos fundamentales para el desarrollo experimental de esta investigación.

Resumen

Una experiencia inmersiva puede definirse como el resultado sensorial, perceptual y emocional que experimenta un usuario al interactuar con un sistema capaz de generar una sensación de presencia en un entorno artificial. Una inmersión compartida, en contraste, se refiere a la posibilidad de mantener experiencias inmersivas de manera conjunta entre múltiples usuarios, ya sea en espacios locales o remotos, de forma simultánea, síncrona o asíncrona.

Actualmente, la colaboración inmersiva enfrenta exigencias técnicas que se intensifican en entornos con dispositivos heterogéneos, donde las diferencias en capacidades de seguimiento, visualización e interacción pueden comprometer la coherencia de la experiencia. Aunque en la literatura se han propuesto soluciones a estos retos, la mayoría se basan en arquitecturas centralizadas que, si bien son funcionales, presentan limitaciones críticas que nos obligan a plantearnos la pregunta: ¿Deberían las arquitecturas centralizadas seguir siendo el eje principal de la colaboración inmersiva?

Esta obra plantea que un paradigma de colaboración descentralizado podría superar dichas limitaciones, ofreciendo mejoras en la latencia, la robustez y la integración multimodal. Para validar esta hipótesis, se diseñaron, implementaron y evaluaron dos arquitecturas distribuidas para la colaboración entre sistemas de realidad aumentada y realidad virtual: una centralizada y otra descentralizada. Las pruebas se realizaron bajo las mismas condiciones en cuanto a latencia, replicación y coherencia espacial entre dispositivos heterogéneos.

Los resultados obtenidos evidencian el potencial de las arquitecturas descentralizadas como alternativa viable para la colaboración inmersiva, representando una base sólida para futuras investigaciones en sistemas XR.

Abstract

An immersive experience can be defined as the sensory, perceptual, and emotional outcome that a user undergoes when interacting with a system capable of generating a sense of presence within an artificial environment. In contrast, a shared immersion refers to the possibility of sustaining immersive experiences jointly among multiple users, whether in local or remote spaces, simultaneously, synchronously, or asynchronously.

Currently, immersive collaboration faces technical demands that intensify in environments with heterogeneous devices, where differences in tracking, visualization, and interaction capabilities may compromise the coherence of the experience. Although the literature has proposed solutions to these challenges, most rely on centralized architectures which, while functional, present critical limitations that compel us to ask: Should centralized architectures continue to be the main axis of immersive collaboration?

This work argues that a decentralized collaboration paradigm could overcome such limitations, offering improvements in latency, robustness, and multimodal integration. To validate this hypothesis, two distributed architectures for collaboration between augmented reality and virtual reality systems were designed, implemented, and evaluated: one centralized and the other decentralized. The tests were conducted under identical conditions regarding latency, replication, and spatial coherence among heterogeneous devices.

The results obtained highlight the potential of decentralized architectures as a viable alternative for immersive collaboration, representing a solid foundation for future research in XR systems.

Abreviaciones y siglas

Sigla	Término completo	Descripción contextual
RA	Realidad Aumentada	Tecnología que superpone elementos digitales sobre el entorno físico en tiempo real.
RV	Realidad Virtual	Tecnología que genera entornos completamente digitales, inmersivos y simulados.
RM	Realidad Mixta	Integración de RA y RV, donde objetos físicos y virtuales coexisten e interactúan.
XR	Realidad eXtendida	Término paraguas que engloba RA, RV y RM.
HMD	Head-Mounted Display	Dispositivo montado en la cabeza que permite visualizar entornos virtuales o aumentados.
IP	Internet Protocol	Protocolo que identifica y direcciona dispositivos en una red para el envío de paquetes.
SDK	Software Development Kit	Conjunto de herramientas y librerías para desarrollar aplicaciones específicas.
P2P	Peer-to-Peer	Arquitectura de red donde los dispositivos se comunican directamente sin una entidad central.
TCP	Transmission Control Protocol	Protocolo de transporte con conexión, garantizando entrega ordenada y direccionamiento.
UDP	User Datagram Protocol	Protocolo de transporte sin conexión, rápido pero sin garantía de entrega, ideal para datos sensibles al tiempo.

Índice general

Índice de figuras

Capítulo 1

Introducción a la colaboración inmersiva

En este capítulo se presenta un panorama general de la Realidad Aumentada (RA) y la Realidad Virtual (RV) como sistemas colaborativos. Se examinan los antecedentes más representativos que permitieron la creación y la evolución de este campo, destacando los principales retos que enfrenta la nueva generación de este tipo de experiencias. Posteriormente, se sugiere la creación de un nuevo marco conceptual de colaboración, el principal aporte de este trabajo. Como cierre, se resume la organización general de la tesis, señalando cómo contribuye cada capítulo al cumplimiento de la hipótesis y los objetivos de esta investigación.

1.1. De experiencias aisladas a la colaboración

La RA y la RV han pasado de ser experiencias aisladas a convertirse en la próxima generación de herramientas colaborativas [1]. Su capacidad de complementar o reemplazar la percepción del entorno físico ha abierto nuevos paradigmas de aprendizaje [2], diseño [3], comunicación y difusión del conocimiento [4]. Sin embargo, en paralelo a su crecimiento, la aparición de ecosistemas inmersivos cada vez más complejos se vuelve evidente, haciéndose necesaria la ubicuidad entre sus distintas modalidades de interacción; una integración de experiencias híbridas entre RA y RV, habitualmente enmarcadas bajo el término de Realidad eXtendida (XR).

XR es un concepto que generaliza la noción de Realidad Mixta (RM), así como todas sus posibles modalidades o variantes, a lo largo del continuo de virtualidad descrito por Milgram [5]. Al extender su campo de aplicación a áreas como la educación [6, 7], la medicina [8], el diseño industrial o la capacitación remota [9], entre otras, se han identi-

ficado limitaciones que afectan la fluidez, la experiencia y la calidad de la colaboración. Para ilustrar estas limitaciones, considérese, como ejemplo, un escenario en el que coinciden usuarios equipados con visores de RV, dispositivos móviles con RA y computadoras de escritorio convencionales. Esta diversidad de dispositivos, con diferencias en capacidad de cómputo, sensado, visualización e interacción, se denomina entorno heterogéneo. Si las disparidades de este entorno no se contemplan en el diseño de un sistema colaborativo, la experiencia compartida entre los usuarios se fragmenta, la colaboración se debilita, además la sensación de presencia y coherencia espacial se reduce [10, 11].

Para este tipo de escenarios híbridos, los desafíos encontrados en la literatura especializada se agrupan generalmente como tres retos centrales a resolver: (i) alcanzar una sincronización precisa entre las plataformas involucradas (colaboración en tiempo real); (ii) garantizar interoperabilidad entre los dispositivos dispares (heterogeneidad); y (iii) diseñar modelos de interacción que puedan sostener tanto configuraciones simétricas como asimétricas (multimodales) [12, 13].

Una solución ampliamente adoptada para abordar estos retos ha sido el uso de arquitecturas centralizadas como núcleo de la colaboración [14, 15, 16]. En este tipo de arquitecturas, un servidor, también conocido como nodo maestro o dispositivo central, se encarga de sincronizar y distribuir los datos entre todos los usuarios, una solución que ha demostrado ser efectiva al mantener la coherencia e intermediar en la resolución de conflictos. Plataformas comerciales como Photon Engine [17], Unity Netcode [18], Mirror [19] o Microsoft Azure PlayFab [20] ofrecen entornos accesibles que facilitan el desarrollo y el despliegue de prototipos colaborativos con este mismo esquema de solución.

No obstante, las soluciones centralizadas suelen presentar algunas limitaciones significativas: concentran el tráfico en un único dispositivo, disponen de una escalabilidad limitada, incrementan los tiempos de respuesta conforme crece el número de usuarios y son vulnerables a sufrir puntos únicos de falla, como la interrupción total del sistema ante la desconexión del nodo maestro encargado de la comunicación. Además, si se depende de soluciones propietarias, se restringe la personalización y apertura tecnológica, condicionando la investigación abierta. Estas condiciones plantean la necesidad de reconsiderar si dicho enfoque debe seguir siendo el eje principal de la colaboración inmersiva, y abren la posibilidad de explorar modelos alternativos más rápidos, consistentes y robustos para entornos XR heterogéneos.

Durante el resto de este trabajo se profundizará en la conceptualización, implementación y validación de un modelo de comunicación descentralizado, diseñado para responder a los desafíos identificados en entornos colaborativos de XR. A través de este modelo, construido bajo la filosofía de código libre, se mostrará una comparativa sólida entre estos dos tipos de arquitecturas de colaboración inmersiva: arquitectura centralizada vs. arquitectura descentralizada, con la que se permita dar solución a las condiciones anteriormente planteadas, asentando las bases de futuras líneas de investigación en interacción multimodal, coordinación en tiempo real y diseño de sistemas XR colaborativos.

1.2. Antecedentes históricos de la colaboración XR

Una de las primeras iniciativas documentadas en la literatura sobre inmersión colaborativa fue *Cave Automatic Virtual Environment* (CAVE) [21]. En el sistema CAVE, se acondicionó un entorno rectangular para proyectar imágenes estereoscópicas directamente sobre superficies físicas, como los muros y el suelo. Esta configuración permitió generar una sensación de inmersión sin recurrir a dispositivos de visualización individual como los visores HMD (*Head-Mounted Display*), lo que facilitó la interacción simultánea de varios participantes en el mismo espacio compartido.

Aunque CAVE supuso un hito relevante en la evolución de la colaboración inmersiva en entornos presenciales, el proyecto *Distributed Interactive Virtual Environment* (DIVE) [22] presentó una perspectiva diferente: la colaboración a distancia. En DIVE se desarrolló una plataforma multiusuario en la cual los participantes podían interactuar virtualmente, accediendo desde estaciones independientes con configuraciones heterogéneas. Esta diferencia en su arquitectura marcó un punto de inflexión en el desarrollo de las experiencias colaborativas en XR, al pasar de experiencias colocalizadas a escenarios remotos que exigían nuevos mecanismos de sincronización, replicación y coherencia entre entidades virtuales.

Esta transición no fue instantánea, sino que se desarrolló mediante diversas aproximaciones complementarias dentro del continuo de modalidades XR, resumiendo las más relevantes en la Figura 1.1. Por ejemplo, Macedonia y Zyda [23] implementaron NPSNET-IV, un mundo virtual con elementos tridimensionales que incorporó el protocolo IEEE 1278 DIS [24] junto con la difusión mediante multicast IP, tecnologías adoptadas en contextos de entrenamiento militar. En cambio, Greenhalgh y Benford [25] propusieron MASSIVE, un esquema de consistencia espacial, orientado a procesar los componentes gráficos en función de su proximidad dentro del espacio digital.

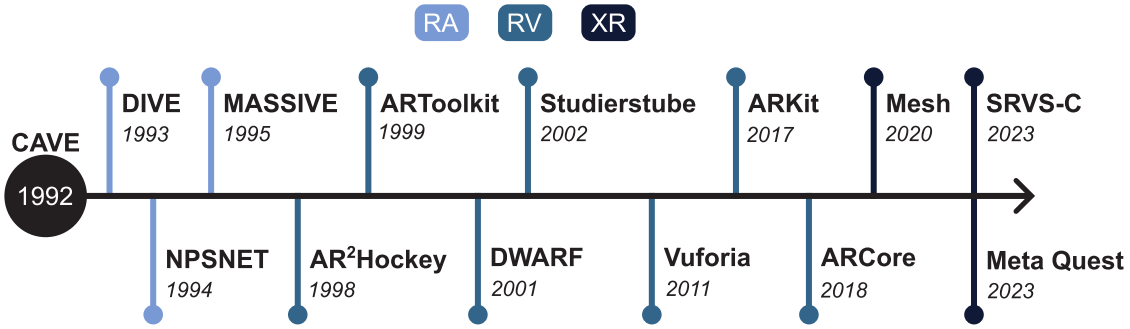


Figura 1.1: Línea del tiempo con los hitos más relevantes en la evolución de las experiencias XR colaborativas a lo largo de los últimos treinta años.

A comienzos de los años 2000, el campo de la RA dio lugar a estudios relevantes como AR²Hockey, por Ohshima *et al.* [26], que evidenciaron la posible sincronización entre elementos virtuales y objetos reales. En paralelo, ARToolkit contribuyó al desarrollo de aplicaciones colaborativas mediante técnicas de mapeo visual distribuido en múltiples disposi-

tivos [27]. En cambio, *Distributed Wearable Augmented Reality Framework* (DWARF)[28], presentado por Bauer *et al.* , introdujo una estructura más formalizada, proponiendo una arquitectura portátil compuesta por módulos remotos. Así mismo, el sistema de estudio enriquecido Studierstube de Schmalstieg *et al.* [29], propuso emplear proyección y visualización tridimensional para el análisis colocalizado bajo una configuración heterogénea, propuesta que adelantó los principios de la interacción ubicua y colaboración local entre múltiples usuarios.

Pese a que estos aportes sucedieron en un mismo periodo, cada uno adoptó enfoques particulares que demostraron la viabilidad de la colaboración entre este tipo de entornos. Plataformas como Vuforia Engine [30], Apple ARKit [31] y Google ARCore [32], también impulsaron su adopción al integrar funciones que hoy resultan fundamentales en la creación de experiencias aumentadas entre las que se destacan el seguimiento espacial, la persistencia y la sincronización entre usuarios [33, 34]. En años más recientes, plataformas como Microsoft Mesh [35] y Spatial Systems [36], así como los SDKs (*Software Development Kits*) Meta XR Core [37], Apple RealityKit [38], Mixed Reality Toolkit [39] y Unity XR [40], han impulsado el diseño de arquitecturas híbridas que combinan computación en la nube y representación espacial compartida, habilitando modalidades avanzadas de interacción en ecosistemas XR.

No obstante, aunque la colaboración centralizada se considera el modelo de comunicación por excelencia, aún son escasos los trabajos que exploran sistemas desarrollados bajo este paradigma. Como antecedente del presente trabajo, se cuenta con una tesis de licenciatura defendida en 2023 [41], la cual abordó la colaboración aumentada entre dispositivos móviles. Esta investigación sentó las bases para el desarrollo de SRVS–C (*Spatially Referenced Virtual Synchronization for Collaboration*) [42], un primer modelo de colaboración híbrido entre RA y RV, cuya implementación permite examinar con mayor detalle las limitaciones técnicas que, abordadas en extenso en el Capítulo 3, sustentan la necesidad de replantear las aproximaciones actuales y explorar nuevos esquemas colaborativos.

1.3. Hipótesis

Gran parte de las soluciones adoptadas en el diseño de ambientes inmersivos compartidos recurren a paradigmas de red centralizados. Aunque una solución centralizada es más que suficiente en múltiples escenarios, se pone en discusión su adopción como eje central en la colaboración entre este tipo de entornos. Ante esta marcada tendencia, se plantea que un paradigma de colaboración descentralizado podría ofrecer mejoras considerables, manifestadas en un menor tiempo de sincronización, mayor robustez frente a errores, e incremento en la capacidad de integración en tareas colaborativas sin afectar la percepción espacial ni el origen de la interacción. De esta manera, para la validación de esta hipótesis, se establecen los siguientes objetivos que permitan evaluar técnica y conceptualmente el modelo sugerido.

1.4. Objetivo general

Diseñar, implementar y someter a evaluación un esquema de comunicación distribuido y multimodal, orientado a entornos colaborativos que integren dispositivos heterogéneos en escenarios virtuales y aumentados.

1.4.1. Objetivos específicos

- Diseñar una arquitectura descentralizada que garantice la sincronización precisa de los estados espaciales entre soluciones basadas en RA y RV.
- Proponer un esquema compacto y multiplataforma para la representación de datos que facilite la transmisión de ambientes virtuales.
- Someter a validación una arquitectura descentralizada mediante métricas cuantitativas, evaluando parámetros como la latencia, la coherencia del estado compartido y la percepción de continuidad en la experiencia colaborativa.

1.5. Delimitación y alcance de la investigación

Debido a la naturaleza multidisciplinaria y el grado de complejidad técnica del presente trabajo, se han definido los siguientes criterios para delimitar el alcance de esta investigación:

Tecnología: La fase experimental se ha centrado en dispositivos móviles compatibles con Google ARCore [43] y en unos visores Oculus Rift con un par de controladores de seis grados de libertad (6DoF). Se excluye el uso de visores de última generación (HoloLens 2, Meta Quest 3, Apple Vision Pro), así como de sistemas completamente vestibles o de cualquier configuración que dependa de sensores externos de alta precisión.

Evaluación experimental: Las pruebas se desarrollaron sobre un espacio de escala de habitación (3×3 metros), utilizando métricas como el retraso en la sincronización y el tiempo que tarda la replicación de una operación. No se incluyen un análisis del consumo de recursos computacionales ni se realizan pruebas con usuarios finales o condiciones operativas reales.

Paradigma arquitectónico: Se contrastan dos modelos de colaboración: una arquitectura centralizada y una arquitectura descentralizada, ambas desarrolladas mediante librerías de código libre para asegurar el control completo sobre las implementaciones. Dado que no existen modelos abiertos fieles a los objetivos de esta investigación, se excluyen comparaciones con soluciones comerciales, plataformas propietarias o esquemas híbridos que dependan de servidores externos.

Componentes funcionales: La investigación se enfoca en cuatro módulos clave para la comparación: (i) sincronización de estados espaciales (posición y orientación), (ii) gestión de sesiones multiusuario, (iii) intercambio de entradas multimodales (movimientos, gestos, eventos), y (iv) visualización coherente del entorno compartido.

1.6. Organización de la tesis

Con el propósito de presentar de forma lineal los aspectos teóricos, técnicos y experimentales vinculados a esta propuesta de descentralización inmersiva, esta tesis se organiza en seis capítulos. A continuación, se resume brevemente el contenido que se podrá encontrar en cada uno de ellos:

Capítulo 1 – Introducción. Presentó el marco general de la investigación, incluyendo los antecedentes tecnológicos y conceptuales, la justificación del estudio, la formulación de la problemática, los objetivos, la hipótesis, las delimitaciones del proyecto y la organización del documento.

Capítulo 2 – Fundamentos teóricos. Se introducen los conceptos técnicos necesarios para comprender la propuesta, que contempla principios de entornos distribuidos, sincronización en redes, tolerancia a fallos y modelos de interacción multimodal.

Capítulo 3 – Estado del arte. Expone un análisis crítico de la literatura especializada sobre arquitecturas colaborativas para sistemas XR. Se examinan las restricciones que presentan los enfoques centralizados y se identifican los vacíos que dan origen a esta propuesta.

Capítulo 4 – Implementación. Documenta detalladamente el diseño arquitectónico del sistema, que incorpora diagramas, estructuras y esquemas de comunicación. Asimismo, se introduce el proceso de implementación del sistema.

Capítulo 5 – Metodología y resultados. Describe la metodología empleada para la evaluación de la propuesta. Se detallan las especificaciones técnicas, los instrumentos empleados, los criterios de evaluación y el protocolo experimental definido. Posteriormente se presentan las pruebas realizadas y el análisis comparativo de los resultados obtenidos.

Capítulo 6 – Conclusiones y trabajo futuro. Sintetiza los resultados más relevantes de la investigación, analizando los aspectos técnicos y conceptuales de ambas arquitecturas. Posteriormente, se recapitulan los objetivos de este trabajo y se proponen líneas de investigación y trabajos futuros orientados a la mejora en la colaboración XR.

Capítulo 2

Fundamentos para la descentralización en entornos XR

En este capítulo se presenta el contenido técnico necesario para comprender los fundamentos de la computación distribuida, abordando sus principios operativos, las arquitecturas más representativas y diversos modelos de comunicación, replicación, consistencia y sincronización de estados. El capítulo concluye con una revisión de las principales herramientas utilizadas en la creación de experiencias XR.

2.1. Introducción a la colaboración inmersiva

La evolución de la colaboración inmersiva ha estado marcada por dos enfoques predominantes: la realidad extendida y el concepto de metaverso. Este último, concebido inicialmente como una visión futurista de interacción social en entornos virtuales, apareció por primera vez en la novela *Snow Crash* de Neal Stephenson [44], una historia distópica que sucede en un extenso mundo virtual que coexiste con el mundo físico. En 2021, la empresa Meta retomó esta noción, atrayendo el interés de medios, industrias tecnológicas y usuarios al materializar un concepto que hasta entonces pertenecía a la ficción [45].

La XR, en cambio, constituye un modelo conceptual orientado al diseño de entornos interactivos. Representa un conjunto tecnológico amplio que busca integrar de manera fluida el espacio físico con escenarios completamente virtuales [46]. Su evolución ha estado marcada por la convergencia de modalidades como la realidad aumentada, que proyecta

elementos digitales sobre el mundo real [47]; la realidad virtual, que introduce al usuario en escenarios tridimensionales generados por computadora [48]; y la realidad mixta, que permite la cohabitación e interacción en tiempo real entre objetos físicos y digitales.

Esta interacción se enriquece mediante el uso de interfaces multisensoriales de entrada y salida que integran sentidos y canales de difusión de manera simultánea [49], buscando emular los patrones de la comunicación humana [50]. Además de los dispositivos de entrada clásicos (pantallas táctiles y controladores), se han integrado progresivamente tecnologías de reconocimiento de comandos por voz y diálogos [51], reconocimiento de gestos faciales, de manos [52, 53] y corporales para la manipulación directa de objetos [54], seguimiento ocular que permite deducir la intención del usuario y su punto de interés [55], así como retroalimentaciones visuales, sonoras, olfativas, gustativas o electro estimulantes (hápticas) como medios de salida para proporcionar una sensación de tacto y presencia física [56, 57].

Cuando estos principios se extienden a dinámicas grupales, se produce entonces una colaboración inmersiva multimodal. Pero esta capacidad de crear espacios compartidos y coherentes no es solo una cuestión de integración de hardware de ultima generación. Representa un reto técnico de alta complejidad de ingeniería de software que se remite a la computación distribuida. En consecuencia, analizar sus restricciones, arquitecturas y soluciones técnicas que permiten la colaboración hace indispensable una familiarización con sus principios operacionales.

2.2. Principios de la computación distribuida

Se entiende por sistema distribuido un conjunto de computadoras autónomas que cooperan entre sí para ofrecer una experiencia unificada y coherente al usuario [58]. El concepto de "usuario" abarca tanto personas como aplicaciones y dispositivos que acceden a los servicios del sistema. La clave de este modelo es que la comunicación y la colaboración entre sus componentes ocurren de forma oculta, de modo que las complejidades técnicas permanecen invisibles para los usuarios del sistema.

Estos sistemas se construyen con el propósito de ofrecer un acceso eficiente, controlado y confiable a recursos que se encuentran úbicados físicamente en distintos lugares. Dichos recursos abarcan desde el almacenamiento, capacidad de cómputo, bases de datos, hasta elementos como impresoras, servicios multimedia o redes de comunicación. Su distribución no solo optimiza el uso de los recursos y reduce costos operativos, sino que también fomenta la colaboración y la interconexión a gran escala, siendo Internet el ejemplo más representativo al posibilitar el intercambio de mensajes, archivos, aplicaciones y contenidos audiovisuales entre millones de usuarios.

Existen dos niveles complementarios que permiten comprender a un sistema distribuido. El primero es el nivel físico, relacionado con el hardware que contempla la disposición real

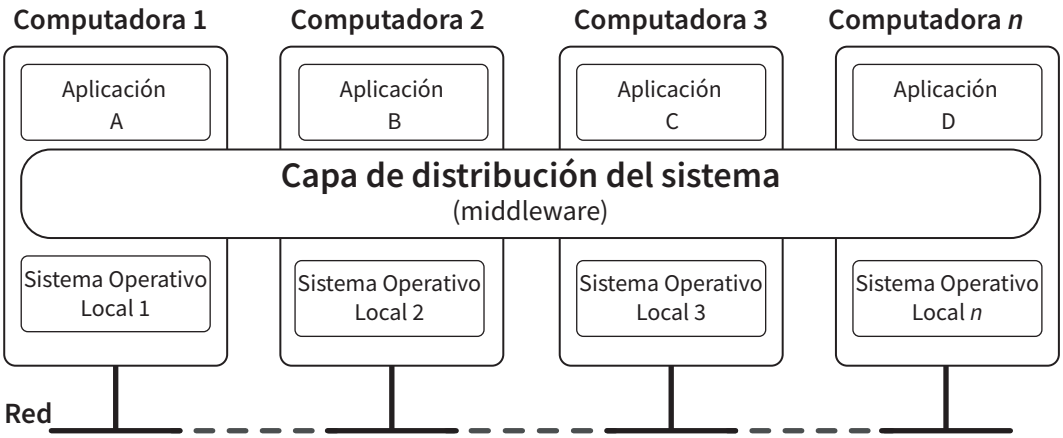


Figura 2.1: Representación de la distribución de un sistema como una capa de middleware. Imagen adaptada de Tannenmbaum [58].

de los dispositivos (servidores, estaciones de trabajo, sensores) y la infraestructura de comunicación que los interconecta. El segundo es el nivel lógico, asociado al software, que constituye el enfoque principal de este trabajo. En este nivel se abstraen los detalles físicos para definir cómo los componentes se comunican, cooperan y se integran para cumplir los objetivos del sistema, lo cual se materializa mediante arquitecturas de software y reglas de interacción bien establecidas entre los usuarios.

Una característica esencial de los sistemas distribuidos es su autonomía: permite actualizar, reemplazar o reconfigurar partes del sistema sin interrumpir su operación global. Si se visualiza a un sistema distribuido como un sistema compuesto por distintas capas, donde la capa inferior corresponde al sistema operativo y la capa superior a las aplicaciones, tal y como se ilustra en la Figura 2.1, podría introducirse una capa intermedia entre ambas para gestionar la comunicación entre las aplicaciones y el sistema operativo. Esta capa intermedia de distribución, también conocida como *middleware*, idealmente debería cumplir con tres propiedades fundamentales:

1. Transparencia: Oculta la complejidad del sistema al usuario.
2. Apertura: Garantiza la interoperabilidad mediante estándares.
3. Extensibilidad: Permite la integración de nuevos componentes sin comprometer la operación general.

No obstante, diseñar este tipo de sistemas no se reduce únicamente a conectar componentes autónomos bajo una lógica común. En la práctica, se enfrentan retos relacionados con la sincronización de procesos, la consistencia de los datos, la robustez y la seguridad. A continuación, se analizan las arquitecturas y modelos más representativos que permiten materializar dichos principios.

2.3. Arquitecturas distribuidas

Una arquitectura representa el esquema estructural sobre el cual se construye un sistema, definiendo sus componentes, mecanismos y reglas de comunicación e interacción [59]. En un sistema distribuido, esta organización se divide en tres variantes principales: centralizada, descentralizada e híbrida.

2.3.1. Arquitectura centralizada

En una arquitectura centralizada, la coordinación y el intercambio de recursos se realizan mediante un dispositivo o nodo principal, comúnmente denominado servidor. Los demás nodos, denominados clientes, realizan peticiones a este servidor para comunicarse con el resto de los dispositivos, siendo el modelo cliente-servidor el ejemplo más representativo.

En este modelo, la interacción sigue el patrón petición-respuesta: el cliente envía un requerimiento con los datos necesarios, posteriormente el nodo servidor procesa y devuelve una respuesta. Su adopción se debe a su simplicidad, centralizar la lógica de comunicación permite controlar, administrar y asegurar la coherencia de los datos desde un único dispositivo.

2.3.2. Arquitectura descentralizada

En cambio, una arquitectura descentralizada distribuye tanto los recursos como las responsabilidades de coordinación entre los demás dispositivos. En esta arquitectura, los nodos actúan simultáneamente como emisores y receptores, compartiendo recursos y colaborando de manera directa sin necesidad de un dispositivo principal. Esta descentralización, sin embargo, introduce una mayor complejidad en la lógica de sincronización, especialmente en operaciones que requieren coherencia espacial, replicación simultánea o resolución de conflictos en tiempo real. Las redes *peer-to-peer* (P2P) son el ejemplo más representativo de este esquema.

Esta arquitectura ofrece una mayor robustez ante errores, ya que el sistema no se ve comprometido por la caída, el corte o la suspensión de un nodo. Así mismo, facilita la escalabilidad, pues los recursos disponibles crecen al incorporar nuevos nodos. Aplicaciones de comunicación descentralizada como Skype en sus primeras versiones [60], o infraestructuras más recientes como la blockchain [61] usan una implementación directa de este tipo de arquitecturas.

2.3.3. Arquitectura híbrida

Las arquitecturas híbridas integran características de los modelos centralizados y descentralizados, con el fin de aprovechar sus ventajas y cubrir sus limitaciones. En las arquitecturas híbridas, ciertos nodos coordinan tareas específicas, mientras que otros operan de forma autónoma y colaborativa. Esta arquitectura resulta especialmente útil en entornos dinámicos o heterogéneos, donde los nodos cuentan con capacidades y recursos dispares con distintos objetivos y tareas, siempre y cuando su implementación separe los mecanismos centralizados y descentralizados.

2.4. Protocolos de comunicación

Independientemente de la arquitectura lógica empleada en un sistema distribuido, los entornos XR colaborativos enfrentan uno de sus desafíos más exigentes: la coherencia espacial. Crear la ilusión de un espacio compartido y coherente se reduce a la capacidad de una arquitectura de red para sincronizar el estado del mundo virtual entre múltiples participantes con el menor retraso posible en la actualización de este.

En el núcleo de toda comunicación en Internet, y por extensión en aplicaciones XR colaborativas que dependen de redes IP (*Internet Protocol*), se encuentra la familia de protocolos TCP/IP (*Transmission Control Protocol/Internet Protocol*)[62]. Este modelo estandarizado organiza la comunicación en cinco capas que, trabajando de manera concertada como se visualiza en la Figura 2.2, permiten el transporte de datos desde la aplicación hasta el medio físico y viceversa.

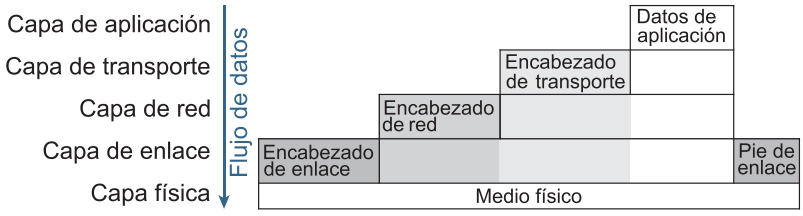


Figura 2.2: Representación de la distribución en capas del protocolo TCP/IP. Imagen adaptada de Glazer [62].

2.4.1. Envío de mensajes a través del protocolo TCP/IP

El proceso de transmisión de datos a través de la familia de protocolos de comunicación TCP/IP puede entenderse como la construcción progresiva de un mensaje que atraviesa cada capa de este modelo de arquitectura de red. Cada nivel añade información especí-

fica para su correcta entrega [63]. Cuando una aplicación necesita enviar datos, inicia el siguiente proceso de encapsulamiento.

En la capa de aplicación, los datos generados se estructuran según el formato deseado por la aplicación destino. Estos datos pueden incluir información como el tipo de mensaje, tiempo de generación, identificadores, propiedades, así como cualquier otro dato de interés, conformando el segmento de aplicación.

Este primer segmento pasa a la capa de transporte, donde según el protocolo de transporte elegido, típicamente TCP (*Transmission Control Protocol*) o UDP (*User Datagram Protocol*), se encapsula con una cabecera adicional. En el caso de UDP, la cabecera añade el puerto de origen y de destino, la longitud del mensaje y la validación del mensaje (*checksum*). Mientras que, de usarse TCP, se añadirían además números de secuencia y acuses de recibo, formando con ello el segmento de transporte.

El segmento de transporte llega a la capa de red, cuyo elemento fundamental es la dirección IP, un identificador numérico único que distingue cada dispositivo conectado a la red. La asignación de una dirección IP puede ser estática (configurada manualmente para dispositivos críticos como servidores) o dinámica, que asigna y cambia automáticamente direcciones disponibles a los dispositivos que se conectan. En esta misma capa, se encapsula una cabecera IP que contiene las direcciones IP de origen, de destino, el protocolo de transporte utilizado, mecanismos para prevenir bucles infinitos y otros campos de control. Esta nueva encapsulación forma el paquete IP, que ahora puede ser compartido a través de múltiples redes intermediarias.

En la capa de enlace de datos, el paquete IP se encapsula dentro de una trama específica del medio físico (Ethernet, Wi-Fi, etc). Se añaden las direcciones MAC (*Media Access Control*) del dispositivo origen, información de control de acceso al medio y una secuencia de verificación de la trama de datos para la detección de errores, formando la trama de enlace.

Finalmente, en la capa física, la trama de enlace, que integra todas las demás tramas de información realizadas hasta el momento, se convierte en señales eléctricas, electromagnéticas u ópticas según el medio de transmisión (cable coaxial, fibra óptica, ondas de radio), listas para ser transmitidas a través del medio físico. Aunque comúnmente las capas más relevantes que se manipulan son las de transporte (responsable de la comunicación de extremo a extremo) y la capa de aplicación (donde residen los protocolos específicos de la colaboración), la comprensión integral de toda la familia de protocolos de comunicación TCP/IP resulta ser clave para trabajos de optimización y rendimiento.

2.4.2. El dilema del transporte: TCP frente a UDP

La elección del protocolo de transporte define el estándar de comunicación que se usará entre los componentes de un sistema [64]. TCP es un protocolo orientado a la conexión que garantiza la entrega ordenada y sin errores de todos los paquetes. Esto se consigue mediante acuses de recibo y retransmisiones automáticas en caso de pérdidas. Esta confidencialidad lo hace ideal para datos de control y estado críticos que deben llegar intactos, como la lógica de aplicación, la creación o destrucción de objetos o la sincronización de un estado persistente. Sin embargo, sus mecanismos de control y retransmisión introducen retrasos variables que son considerables para la sincronización en tiempo real.

UDP, en cambio, envía paquetes (llamados datagramas) sin garantía de entrega, orden o integridad. Esta falta de sobrecarga lo hace más rápido y con retardos significativamente bajos, lo que lo hace adecuado para la transmisión continua de datos sensibles al tiempo, como operaciones de seguimiento, transmisiones de voz o actualización de avatares. Es decir, cualquier dato obsoleto que resulte menos valioso que un nuevo dato actual, es preferible descartar su perdida y esperar la siguiente actualización que esperar su retransmisión. Dadas estas diferencias, es habitual que las arquitecturas modernas empleen distintos protocolos y adopten enfoques híbridos.

Tabla 2.1 Dominios de comunicación en sockets Berkeley

Dominio	Descripción
`AF_INET`	Comunicación sobre IPv4, utilizada en arquitecturas cliente-servidor convencionales.
`AF_INET6`	Comunicación sobre IPv6, adecuada para entornos modernos con direccionamiento extendido.
`AF_UNIX`	Comunicación local entre procesos mediante archivos de socket, útil para pruebas.
`AF_PACKET`	Acceso directo a nivel de enlace (Ethernet).
`AF_NETLINK`	Canal de comunicación entre el kernel y el espacio de usuario, utilizado en configuraciones avanzadas sobre Linux.
`AF_CAN`	Redes de área de controlador, común en aplicaciones automotrices.
`AF_BLUETOOTH`	Comunicación inalámbrica entre dispositivos Bluetooth.
`AF_VSOCK`	Comunicación entre máquinas virtuales, empleada en entornos virtualizados o contenedores.
`AF_TIPC`	Comunicación distribuida en sistemas de alta disponibilidad.

2.4.3. Sockets Berkeley: Interfaz de programación para comunicación en red

La interacción entre estas capas del protocolo de comunicación TCP/IP se realiza programáticamente a través de interfaces conocidas como sockets Berkeley [65], un estándar que proporciona una abstracción unificada para la comunicación entre procesos, ya sea localmente o a través de la red. Desarrollado originalmente en la Universidad de California, Berkeley, como parte de los sistemas BSD UNIX, este modelo se ha convertido en la base de la programación de redes en la mayoría de los sistemas operativos modernos.

Un socket se conceptualiza como un punto final abstracto para el envío y la recepción de datos, funcionando como un canal bidireccional de comunicación entre dos usuarios. Desde la perspectiva de un programador, un socket se comporta de manera similar a un descriptor de archivo, permitiendo operaciones de lectura y escritura mediante llamadas al sistema estándar. Cada socket se caracteriza por tres atributos fundamentales: el dominio de la comunicación (también llamado familia de direcciones, pudiendo ser alguna de las listadas en la Tabla 2.1), el tipo de socket (que define el modelo de comunicación TCP o UDP), y el protocolo específico a utilizar (ver Tabla 2.2).

Tabla 2.2 Tipos de socket y protocolos específicos en entornos XR

Tipo de socket	Descripción	Protocolos comunes
`SOCK_STREAM`	Orientado a la conexión. Proporciona un flujo continuo y confiable de datos. Ideal para sincronización precisa y estados persistentes.	TCP (Transmission Control Protocol)
`SOCK_DGRAM`	Sin conexión. Envía datagramas independientes sin garantía de entrega ni orden. Útil para datos sensibles al tiempo.	UDP (User Datagram Protocol)
`SOCK_RAW`	Permite acceso directo a protocolos de red subyacentes. Usado para diagnóstico o implementación personalizada de protocolos.	IP, ICMP, protocolos definidos por el usuario
`SOCK_SEQPACKET`	Similar a `SOCK_STREAM`, pero preserva los límites de los mensajes. Menos común en XR.	SCTP (Stream Control Transmission Protocol)
`SOCK_RDM`	Proporciona mensajes confiables sin conexión. Poco implementado en sistemas modernos.	Protocolos experimentales o propietarios

Existen dos tipos de sockets ampliamente utilizados: sockets TCP y sockets UDP. Los sockets TCP (tipo `SOCK_STREAM`) proporcionan un flujo de datos continuo y confiable, garantizando la entrega ordenada de los paquetes mediante retransmisiones automáticas. Este modelo es análogo a una llamada telefónica: una vez establecida la conexión, los datos fluyen secuencialmente hasta que alguna de las partes cierra la comunicación. En cambio, los sockets UDP (tipo `SOCK_DGRAM`) operan mediante datagramas independientes, donde cada mensaje constituye una unidad de información autónoma sin garantías de entrega u orden. Esta aproximación se asemeja al servicio postal, donde cada carta viaja de forma independiente y puede perderse o llegar en desorden.

2.5. Serialización y formatos de mensajes

En la Sección 2.4.1 se mencionó que la capa de aplicación se encarga de formatear los datos a transmitir según un esquema común entre los dispositivos involucrados. Este proceso de convertir una estructura de datos, desde su formato presente en la memoria aleatoria, en una secuencia lineal de bits, recibe el nombre de serialización y es independiente del equipo que lo genera [66]. Los objetos serializados pueden ser almacenados, transmitidos y reconstruidos según sea necesario empleando su proceso inverso, la deserialización.

Existen dos enfoques predominantes en este proceso: la serialización binaria y la serialización en texto. En el primer enfoque los objetos se convierten en una secuencia de bytes, esto los hace eficientes en términos de espacio y velocidad. Algunos de estos ejemplos de serialización son: MessagePack que usa códigos de "tipo" y "longitud" para representar datos más rápido y más pequeños que JSON. Protocol Buffers que requiere compilar previamente esquemas de tipo "proto" donde se define la estructura de datos que se desea serializar. FlatBuffers que también requiere compilar previamente esquemas de tipo "fbs", pero que no requieren un proceso de deserialización. Apache Avro que emplea esquemas en formato JSON para especificar la estructura de datos a serializar, pero sin necesidad de compilarla previamente al incluir dicho esquema de interpretación dentro de la secuencia de bytes.

En cambio, la serialización en texto representa los datos en formatos legibles por humanos, lo que facilita la inspección y edición manual. Aunque suelen ser menos eficientes en términos de tamaño y velocidad, son ampliamente adoptados en aplicaciones web y servicios interoperables. Los formatos más comunes son: XML (lenguaje de marcado extensible) un lenguaje de marcado similar a HTML pero sin etiquetas integradas. JSON un formato de serialización de datos ampliamente extendido entre plataformas y lenguajes para intercambiar estructuras del tipo "campo:valor" entre aplicaciones y servicios. YAML un formato enfocado en la legibilidad humana que utiliza una sintaxis basada en la indentación para representar datos estructurados como listas, mapas o valores escalares, siendo especialmente útil en archivos de configuración.

El proceso de la serialización facilita hacer llamadas a otros procedimientos locales o remotos, identificar cambios de datos en ejecución, crear copias de información, añadir persistencia a los datos o intercambiar objetos entre distintos programas. Sin embargo, su uso también acarrea inconvenientes, por ejemplo, rompe la opacidad de los datos abstractos al exponer atributos públicos y privados. Produce retrasos en la transmisión y deserialización si no se garantiza una serialización de objetos optimizada y en caso de serializar objetos que tienen valores que apuntan a direcciones en memoria, al deserializar esta dirección no necesariamente será la deseada.

Aunque existen soluciones particulares que resuelven algunos de estos problemas como la encriptación, compresión u operaciones de referencias lógicas a apuntadores de memoria física (pointer swizzling, lazy swizzling, partial swizzling), el consumo adicional de recursos para implementar alguno de estos procesos puede degradar la experiencia que hace uso de la solución. Por esta razón, ante estructuras mas complejas, se recomienda serializar individualmente por campos y almacenarlos en un buffer previo a su envío, o bien, construir paquetes de información.

2.6. Replicación y consistencia de datos

La replicación de estados se define como el proceso mediante el cuál se mantienen copias sincronizadas del estado del mundo virtual a través de múltiples usuarios. Esta consistencia puede conseguirse bajo un enfoque de consistencia fuerte y eventual. En la consistencia fuerte se garantiza que los estados se vean reflejados en el mismo instante temporal en el que se realiza una actualización. En cambio, en un esquema de consistencia eventual se permite que las réplicas diverjan temporalmente [67].

Un esquema típico de solución suele establecer un dispositivo en la red como la "fuente de la verdad", donde se alberga el estado global del mundo. Cada usuario adicional mantiene una réplica local que se actualiza continuamente. Sin embargo, en un sistema completamente descentralizado, el problema incrementa su complejidad a gran escala cuando se busca que esa sincronización sea convergente, instantánea y robusta entre todos los usuarios.

2.6.1. Mecanismos de control y unicidad

La mayoría de los algoritmos de consenso descritos en la literatura suelen requerir un control respecto a cada nodo involucrado en el sistema. Esta supervisión refuerza la integridad y el carácter único de la información compartida, mediante tres mecanismos comúnmente empleados en este tipo de procesos: marcas de tiempo (timestamp), identificadores UUID y funciones criptográficas aplicadas a los datos [68].

Una marca de tiempo, entendida como una referencia cronológica representada mediante fecha y hora, permite establecer la secuencia de eventos ocurridos dentro de un proceso, resolviendo ambigüedades sobre la prioridad y el orden de ejecución. Por su parte, los UUIDs (*Universally Unique Identifiers*) son identificadores generados mediante algoritmos diseñados para minimizar la probabilidad de colisión, lo que permite referenciar de forma inequívoca las entidades distribuidas en la arquitectura.

La operación de hashing, también conocida como función hash, implica crear una huella digital de la información con una extensión predefinida, utilizada para detectar cambios e inconsistencias entre paquetes transmitidos. Al aplicar funciones hash sobre el estado de los paquetes, el sistema es capaz de identificar divergencias entre réplicas locales y el estado canónico, permitiendo únicamente aquellas modificaciones que hayan sido sincronizadas durante todo el proceso de la comunicación.

2.7. Desafíos técnicos en la colaboración XR distribuida

En aplicaciones tradicionales, un retardo de varios cientos de milisegundos puede resultar aceptable. Sin embargo, los entornos XR exigen latencias inferiores a los 100 milisegundos para preservar la ilusión de presencia [69]. Esta exigencia se ve agravada por el jitter, una fluctuación temporal capaz de alterar la fluidez y consistencia de las interacciones compartidas.

La pérdida de paquetes y la preservación de su integridad añaden una capa técnica adicional que complica la operación del sistema, junto con aspectos vinculados a la privacidad, la protección de datos y la capacidad de crecimiento del sistema [70].

2.7.1. Entornos de programación XR y sus arquitecturas de conectividad

La integración de los principios abordados representa un reto técnico de alta complejidad. Ante esta situación, los motores contemporáneos de creación de experiencias interactivas han incorporado niveles de abstracción que encapsulan los desafíos de red y de sincronización, facilitando que los desarrolladores se enfoquen en la construcción de entornos inmersivos sin afectar la robustez operativa del sistema [71].

Un ejemplo de ello es Unity Engine [72], una plataforma de desarrollo ampliamente utilizada para la creación de experiencias multiusuario mediante estructuras jerárquicas de abstracción. Su sistema nativo de Networking (UNET, actualmente obsoleto) sentó las bases de la conectividad colaborativa, mientras que soluciones más recientes como Netcode

for GameObjects (NGO) [18] ofrecen una estructura de programación de alto nivel que automatiza la replicación de propiedades, la sincronización de escenas y el control de sesiones multiusuario. Para desarrolladores que requieren arquitecturas personalizadas, el *middleware* Photon Engine proporciona una red distribuida alojada remotamente [73], dotada de herramientas como *matchmaking* automático y replicación de estado integradas.

Otro ejemplo es Unreal Engine [74], que aborda la conectividad multiusuario mediante un sistema de replicación embebido de forma nativa en su estructura interna. Mediante el simple marcado de propiedades o mediante el uso de *blueprints* visuales, los desarrolladores pueden indicar qué elementos deben sincronizarse automáticamente a través de la red. El motor gestiona eficientemente la priorización de actualizaciones basándose en la relevancia espacial, replicando primero los objetos más cercanos al jugador. Incluso, para casos de uso empresarial y experiencias a gran escala, Unreal Engine ofrece integración con servicios como Epic Online Services, que proporciona *matchmaking*, logros, almacenamiento en la nube y otras funcionalidades esenciales sin costos de licencia.

En el espacio open-source, Godot Engine [75] ha avanzado significativamente en capacidades de red con su sistema de High-Level Multiplayer API. Aunque requiere más configuración manual que sus contrapartes comerciales, Godot Engine ofrece flexibilidad para implementar arquitecturas personalizadas y sincronización de estados. Su naturaleza abierta permite adaptaciones específicas para requisitos XR, como la optimización para latencia ultrabaja o integración con protocolos especializados.

Si bien estas plataformas constituyen la base tecnológica sobre la cual se construyen experiencias XR colaborativas, su verdadero potencial se revela al analizar implementaciones específicas que han logrado resolver los desafíos de sincronización, escalabilidad y coherencia en entornos distribuidos. El siguiente capítulo aborda estos casos, permitiendo identificar patrones arquitectónicos efectivos y estrategias de diseño replicables.

Capítulo 3

Estado del arte

Este capítulo analiza el estado del arte en torno a los sistemas inmersivos colaborativos, con énfasis en las distintas arquitecturas y modelos de implementación descritos en la literatura especializada. Posteriormente, se desarrolla un estudio comparativo de las distintas soluciones e implementaciones existentes, identificando las características técnicas, los mecanismos de sincronización, así como las fortalezas y limitaciones a las que llegaron distintos autores.

Distintos autores resaltaron tempranamente la importancia de las arquitecturas distribuidas para la creación de experiencias compartidas [76, 77]. Algunos de los avances han sido enfocados en definir marcos puramente conceptuales sobre cómo obtener dicha colaboración, como Schafer *et al.* [78] que definieron el entorno, los avatares y la interacción como los tres componentes fundamentales bajo los cuales se debería implementar la lógica de una experiencia compartida. Mientras que Braud *et al.* [79] identificaron que los sistemas operativos y las arquitecturas de software actuales carecen de las características necesarias para soportar las demandas específicas de la XR, por lo que propusieron integrar soporte nativo de hardware, algoritmos de visión computacional y protocolos de comunicación híbridos como elementos primitivos de un sistema operativo al que llamaron XROS, todo con la finalidad de simplificar el desarrollo de futuras aplicaciones.

Otros autores han optado por evaluar experimentalmente los sistemas distribuidos como en el trabajo de Brown *et al.* [80], quienes presentaron un sistema de distribución de datos basado en eventos para el *Battlefield Augmented Reality System* (BARS), una experiencia en RA diseñada para mejorar la conciencia situacional y coordinación en entornos militares, abordando desafíos como conectividad de red poco confiable, ancho de banda limitado,

protocolos de transporte intercambiables y canales de comunicación. Su trabajo se destacó por ser uno de los primeros en emplear grupos multicast como mecanismo de difusión, demostrando la flexibilidad para adaptarse a diferentes tipos de usuarios y aplicaciones, tanto en RA móvil como sistemas de RV.

3.1. Arquitecturas centralizadas

Existen trabajos que han propuesto nuevas arquitecturas de colaboración. Herskovitz *et al.* [81] introdujeron capacidades valiosas para el reconocimiento espacial compartido en aplicaciones de RA. Su diseño fue exclusivamente adaptado para ecosistemas de dispositivos homogéneos.

Simiscuka *et al.* [82] diseñaron una arquitectura específicamente para abordar los desafíos de sincronización entre dispositivos físicos en el ámbito del Internet de las Cosas (IoT) y entornos virtuales habilitados por la nube. Guo *et al.* [83] desarrollaron Blocks, una aplicación móvil inspirada en la herramienta Google Docs, que permite crear estructuras de RA que persisten en el ambiente físico, contrastando colaboración colocalizada y remota. Por su parte, los trabajos que sugieren nuevos marcos de colaboración, como el de Pereira *et al.* [84] y el de Kostov y Wolfartsberger [85], siguieron un enfoque cliente-servidor. En su defecto, optaron por el uso de frameworks y tecnologías como Photon Engine, SPARQL y Netcode for Gameobject [86].

Las nuevas tendencias se han dirigido hacia los entornos híbridos cloud-edge [87] o sistemas totalmente basados en la nube [88], que ofrecen capacidades de procesamiento y almacenamiento escalables. Pero cuyo rendimiento depende de conexiones a Internet de gran ancho de banda, un requisito poco práctico en escenarios colaborativos locales con infraestructuras limitadas. Una variante interesante la realizaron Viola *et al.* [89], quienes desarrollaron VR2Gather, un sistema de telepresencia que habilitó una comunicación multiusuario en tiempo real mediante la transmisión de contenido volumétrico fotorealista, superando las limitaciones de enviar avatares (uno de los paquetes más complejos de información). O como el trabajo de Han *et al.* [90], quienes propusieron CoMIC (*Collaborative Mobile Inmersive Computing*), una infraestructura para aplicaciones inmersivas geodistribuidas, cuyo aporte incluyó el uso de redes 5G, renderizado remoto de contenido, mapeo espacial compartido y mecanismos de seguridad y privacidad.

El avance tecnológico también ha favorecido la heterogeneidad de las experiencias. Los estudios conceptuales de Numan y Steed [13], junto con Sereno *et al.* [91], articularon cómo las asimetrías tecnológicas se manifiestan bajo dos tipos distintos de colaboración: asimetría perceptiva, donde variaciones en la representación sensorial genera experiencias divergentes, y asimetría interactiva, que surge de disparidades de control e interacción. Brehault *et al.* [92] operacionalizan este marco descomponiendo las asimetrías en ocho dimensiones medibles, proporcionando una matriz de evaluación que conecta desajustes tecnológicos con sus respectivas implicaciones colaborativas.

Sin embargo, contrario a la suposición de que las configuraciones asimétricas podrían ser perjudiciales en una experiencia colaborativa, es decir, que experiencias con el mismo tipo de tecnologías y dispositivos presentan mejores resultados en una experiencia compartida, Grandi *et al.* [93] aportaron evidencia empírica donde demuestran que la interacción asimétrica (heterogénea) entre RA y RV puede ser tan efectiva como configuraciones simétricas (homogénea) del tipo RA–RA o RV–RV. En [10] también mostraron que configuraciones asimétricas entre computadoras personales (PC) y RV incluso superaron configuraciones simétricas en experiencia de usuario y rendimiento de tareas. Agnes *et al.* [94] encontraron que mientras las configuraciones simétricas entre RV–RV generan más ideas (favorecen la creatividad), las asimétricas como emparejamientos PC–RV conducen a la delegación de tareas y comunicación más eficiente. Estos resultados han abierto nuevas oportunidades para la creación de marcos de desarrollo multiplataforma, tal como la propuesta de Huang *et al.* [95] quienes presentaron SCAXR, una arquitectura diseñada para realizar renderizado bajo demanda aprovechando la heterogeneidad computacional entre los dispositivos involucrados.

3.2. Soluciones descentralizadas

Investigaciones recientes también han explorado arquitecturas descentralizadas para XR. Frey *et al.* [96] presentaron Solipsis, una de las primeras arquitecturas completamente descentralizadas para entornos virtuales masivos, que fue diseñada para superar los límites de escalabilidad de los sistemas centralizados tradicionales. Una propuesta basada en tecnologías de Web descentralizada (DWeb) la realizó Huh *et al.* [97], quienes utilizaron la base de datos descentralizada GunDB para sincronizar estados entre pares y un enfoque "offline-first", que permite a usuarios colaborar sin conexión estable. Bajo el mismo paradigma WebXR, en [98] se desarrolló una arquitectura XR descentralizada para entornos de aprendizaje colaborativo basada en el concepto de tiempo virtual, transformando el modelo Croquet, un modelo de computación basado en eventos sincronizados entre pares [99], en una aplicación P2P llamada Luminary que permitió una sincronización sin servidores intermediarios.

Norman *et al.* [100] propusieron una técnica interesante en la que se combinan dispositivos de RA y RV para crear un espacio colaborativo híbrido, donde cada participante puede ver y manipular objetos virtuales desde su propia perspectiva mediante una interacción entre usuarios locales y remotos. Pereira *et al.* [101] presentaron ARENA, una arquitectura XR distribuida basada en cloud–edge que facilita el desarrollo de aplicaciones colaborativas multiusuario sobre navegadores Web, empleando un modelo de escena publicador–subscriptor sincronizado. En estas arquitecturas, los dispositivos actúan como emisores y receptores, demostrando ser viable en la sincronización de interacciones heterogéneas, una propuesta que se consiguió de manera masiva con Decentraland [102], un primer mundo virtual completamente descentralizado que emergió sobre la tecnología blockchain, gobernado por su comunidad mediante una Organización Autónoma Descentralizada (DAO), ofreciendo un ecosistema donde usuarios poseen, crean y controlan sus

activos y experiencias digitales.

Con el crecimiento y popularidad del blockchain, los intentos por explorar oportunidades de incursión con el metaverso se vieron reflejados. Ghosh *et al.* [103], al igual que Jagatheesaperumal *et al.* [104], examinaron detalladamente la relación entre blockchain, la tecnología Web 3.0 y la descentralización del metaverso, coincidiendo en que un enfoque centralizado limitaría el control del usuario, expondría riesgos en la protección de datos y dificultaría la interoperabilidad entre plataformas. Con el propósito de revertir esta tendencia a centralizar las experiencias XR, Huabing *et al.* [105] presentaron CRCDnet, una red descentralizada para visualización distribuida sincrónica en escenarios móviles de RA, construida sobre infraestructura Web y empleando blockchain como mecanismo de indexación para facilitar el intercambio seguro de contenido gráfico procesado y habilitar comunicación directa entre dispositivos (D2D, *Device–to–Device*). Asimismo, en una aplicación orientada al ámbito médico, Shreyansh *et al.* [106] propusieron una arquitectura XR descentralizada para visualización colaborativa de datos tridimensionales, integrando un sistema criptográfico biométrico híbrido, blockchain y almacenamiento distribuido para preservar la confidencialidad de los datos y asegurar la conformidad con estándares regulatorios.

Finalmente, Bhattacharya *et al.* [107] sostienen que la evolución de los sistemas inmersivos basados en realidad aumentada y virtual estará marcada por una transición hacia modelos descentralizados. Estos, impulsados por la convergencia de redes 6G y blockchain, anticipan un ecosistema interoperable donde múltiples plataformas compartan activos, identidades y experiencias sin estar supervisadas por una infraestructura centralizada.

3.3. Vacíos conceptuales identificados en la literatura

La revisión crítica de los trabajos existentes muestra un panorama fragmentado en la evolución de esquemas de colaboración inmersiva en XR. Aunque se observa una tendencia creciente hacia el diseño de experiencias compartidas, subsiste una separación marcada entre los enfoques centralizados predominantes y las propuestas descentralizadas emergentes.

A pesar de que las arquitecturas centralizadas adoptan ampliamente kits de desarrollo de software comerciales como Unity Netcode y Photon Engine, así como enfoques híbridos cloud-edge que cubren las deficiencias en aspectos como la escalabilidad y la conectividad, no se ha conseguido satisfacer las necesidades de colaboración ubicua, siendo pocos los trabajos enfocados en la creación de nuevas arquitecturas. Ante esta oportunidad, se extiende la Tabla 3.1 como un resumen comparativo de las distintas soluciones aportadas a la literatura por diversos autores, en el que la descentralización comienza a destacar como un paradigma prometedor. Por ello proponer una nueva arquitectura que combine los mejores aspectos identificados en cada solución podría marcar los antecedentes de nuevos aportes dentro del campo de la colaboración XR.

Tabla 3.1 Resumen comparativo de artículos sobre colaboración XR

Referencia	Arquitectura	Modalidad	Dispositivos	Interacción
[22] Carlsson 1993	P2P	RV	PC, HMD	Teclado, ratón
[23] Zyda 1994	Distribuida	RV	PC, HMD	Teclado, ratón
[26] Ohshima 1998	Distribuida	RA	HMD	Panel interactivo
[29] Dieter 2002	Distribuida	RA	HMD, PC	Panel interactivo
[80] Brown 2004	Distribuida	RA, RV	HMD, PC	Teclado, ratón, HMD
[96] Frey 2008	P2P	RV	PC, smartphone	Teclado, ratón
[108] Baillard 2017	Cliente-servidor	RA, RM	Tablet, HMD	Táctil, gestos
[109] Dey 2017	Cliente-servidor	RA, RV	HMD	Gestos, avatar
[110] Zhang 2018	Cloud-edge	RA	Smartphone	Táctil
[82] Simiscuka 2019	Híbrida	RV	HMD, PC	Táctil, gestos
[83] Anhong 2019	Cliente-servidor	RA	Smartphone, HMD	Táctil, gestos, voz
[84] Pereira 2019	Cliente-servidor	RA, RV	HMD, smartphone	Táctil, gestos
[97] Huh 2019	P2P	RA, RV, RM	Smartphone, PC, HMD	Táctil, mouse, gestos
[111] Sandor 2019	Cliente-servidor	RA	HMD	Controladores
[34] Kaewrat 2020	Cliente-servidor	RA	Smartphone, PC	Marcadores, táctil
[112] Zenati 2020	Cliente-servidor	RA, RV	Smartphones, HMD	Táctil, gestual
[88] Mourtzis 2020	Cloud	RA	Smartphone, HMD	Táctil
[113] Zhu 2020	Cliente-servidor	RA	Tablets	Táctil
[101] Pereira 2021	Distribuida	RA, RV	Smartphone, HMD, PC	Táctil, gestos, voz
[98] Suslov 2021	P2P	RA, RV	PC, HMD	Teclado, mouse
[33] Manuaba 2021	Cliente-servidor	RA	Smartphone	Táctil
[101] Pereira 2021	Cloud-edge	RA, RV, RM	HMD, PC, smartphone	Gestos, táctil
[14] Lee 2021	Cliente-servidor	RA, RM	HMD, smartphone	Gestos, táctil
[13] Numan 2022	Cliente-servidor	RA, RV	HMD, smartphone	Gestos, táctil
[15] Tumler 2022	Cliente-servidor	RA, RV, RM	HMD, PC	Gestos, táctil
[114] Porcino 2022	Cliente-servidor	RA, RM	HMD, PC, smartphone	Gestos, táctil
[85] Kostov 2022	Cliente-servidor	RA, RV, RM	HMD, PC, smartphone	Controladores, teclado, táctil
[86] Guo 2023	Cliente-servidor	RM	HMD	Gestos, seguimiento ocular
[89] Viola 2023	Híbrida	RV	HMD, smartphone	Voz, movimiento físico
[90] Han 2023	Cloud-edge	RA, RV, RM	HMD, smartphone	Voz, gestos, seguimiento físico y ocular
[115] Huang 2023	Cloud-edge	RA, RV	HMD, smartphone	Gestos, táctil
[95] Huang 2024	Cloud-edge	RA, RV, RM	HMD	Gestos
[116] Neeli 2024	Cloud-edge	RA, RM	HMD, PC, smartphone	Gestos, táctil
[105] Zhang 2024	Descentralizada	RA	Smartphone	Táctil
[106] Sharma 2025	Descentralizada	RA, RV	Smartphone, HMD	Táctil, gestos
[102] Decentraland	Descentralizada	RV	PC	Mouse, teclado

Capítulo 4

Arquitectura, Interfaces e Implementación

Este capítulo presenta el diseño sistemático de dos sistemas colaborativos con sus respectivas interfaces de comunicación. Primero se describe cada módulo de la arquitectura centralizada. Posteriormente, se da espacio para describir el diseño de la arquitectura descentralizada. Finalmente, se presenta la interfaz común del sistema con sus respectivas mecánicas de operación, coherencia espacial y multimodalidad.

4.1. Mecánica general

En esta investigación, se ha adoptado una metodología de experimentación empírica. La propuesta toma como referencia una aplicación de dibujo espacial colaborativo, una modalidad interactiva base entre las aplicaciones inmersivas de acuerdo a Close *et al.* [117]. Esta aplicación, ilustrada en la Figura 4.1, proporciona la capacidad de generar líneas virtuales de distintas complejidades geométricas sobre un lienzo espacial compartido, con el que se evalúan métricas de desempeño como la latencia de transmisión, la replicación y la calidad percibida por el usuario haciendo uso de dos arquitecturas de comunicación opuestas: una arquitectura centralizada y una arquitectura descentralizada.

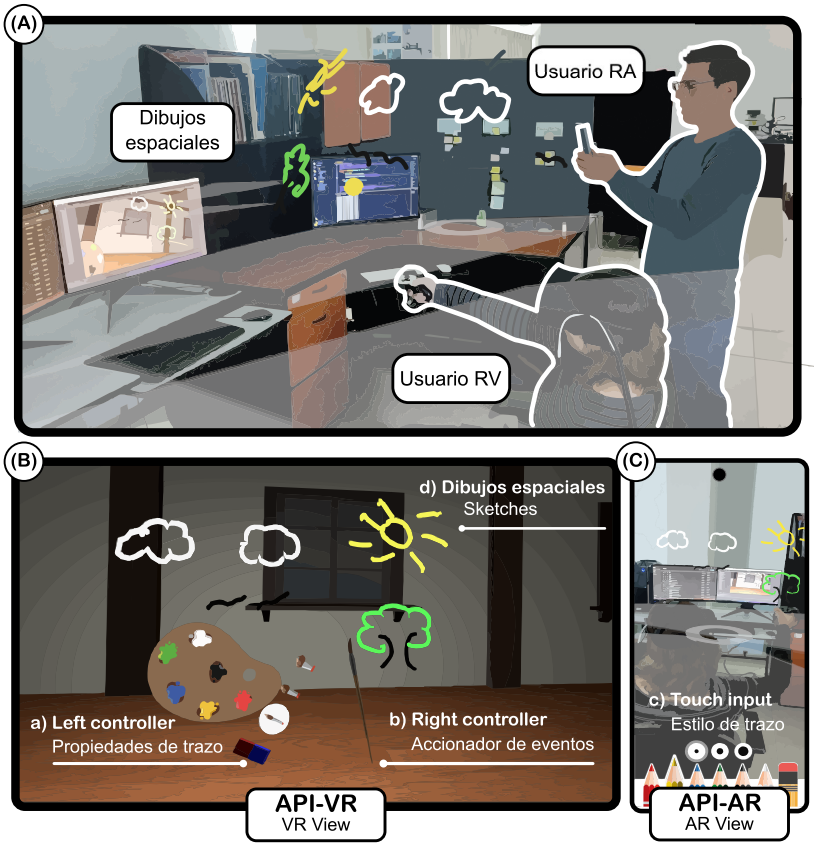


Figura 4.1: Perspectiva de las dos interfaces de usuario en el escenario de dibujo espacial colaborativo. (A) ilustra la disposición espacial de ambos participantes, uno utilizando RA móvil y el otro utilizando RV, junto con sus respectivos trazos virtuales representados en una escena virtual compartida. (B) muestra la interfaz API–VR, donde los usuarios dibujan en un espacio 3D utilizando controladores 6DoF portátiles y ajustan las propiedades del pincel a través de una paleta dentro del mundo virtual. (C) muestra la interfaz API–AR, donde los usuarios dibujan mediante entradas táctiles en un smartphone compatible con ARCore y personalizan los trazos utilizando una barra de herramientas superpuesta en 2D.

Para la implementación, se desarrollaron cuatro prototipos. El primer prototipo corresponde a una arquitectura centralizada. En este diseño, todos los trazos dibujados por los clientes son enviados a un servidor central encargado de retransmitirlos a los demás participantes. En contraste, el segundo prototipo corresponde a una arquitectura descentralizada, diseñada para distribuir la responsabilidad de la comunicación entre los propios participantes. Mientras que el tercer y cuarto prototipo corresponden a la aplicación de dibujo espacial, una orientada a funcionar con RA y la otra con RV respectivamente, que integran a ambas arquitecturas de red desarrolladas para establecer el núcleo de la colaboración y a efecto de su comparación. A continuación, en las secciones siguientes se detalla el diseño de cada uno de los prototipos desarrollados.

4.2. Diseño de la arquitectura centralizada

La arquitectura centralizada, de aquí en adelante referida bajo el acrónimo SRV–C, fue diseñada como un sistema de bloques interconectados para el manejo simultáneo de sesiones colaborativas. Su diseño parte de la necesidad de tener el control total respecto al ciclo de comunicación, pudiendo desplegarse íntegramente en redes locales. Su modelo operativo, resumido en la Figura 4.2, se compone por cinco bloques principales: (i) el bloque de conexiones y desconexiones, (ii) la gestión de usuarios, (iii) el bloque de emparejamiento o *matchmaking*, (iv) el manejador de serialización y (v) el bloque de difusión (Broadcast).

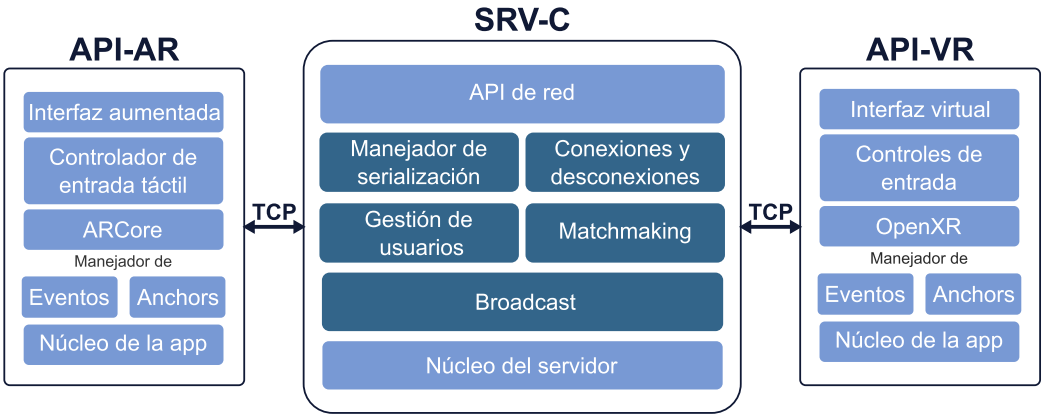


Figura 4.2: Diagrama de composición de la arquitectura centralizada SRV–C.

El núcleo del sistema, mismo que puede replicarse desde el repositorio público en [118], se rige por un modelo cliente-servidor. La comunicación se realiza mediante sockets TCP, implementados sobre una pila tecnológica basada en Node.js, TypeScript y las dependencias descritas en la Tabla 4.1. La información de la sesión colaborativa se almacena en memoria local empleando una estructura de datos, actualizada bajo demanda, para almacenar el estado individual y el estado compartido de cada cliente conectado. Esto facilita el acceso inmediato a la información relevante, sin incurrir en operaciones adicionales sobre bases de datos, mismas que son totalmente opcionales en esta implementación.

4.2.1. Conexiones y desconexiones

El bloque de conexiones y desconexiones gestiona la conexión, desconexión y reconexión de cada dispositivo integrado en el sistema, asegurando que la colaboración, permanencia y eventual recuperación se ejecute de manera mecánica.

Tabla 4.1 Dependencias utilizadas para el desarrollo del servidor con sus respectivos usos

Dependencia	Versión	Descripción
node	22.15.20	Entorno de ejecución para JavaScript del lado del servidor. Se utiliza como base para ejecutar el servidor y manejar procesos en tiempo real.
crypto	10.0.0	Generador de Identificadores Únicos Universales (UUID). Se emplea para crear identificadores únicos en recursos como sesiones, usuarios, o mensajes dentro de la aplicación.
protobufjs	7.5.3	Implementación de Protocol Buffers en JavaScript. Permite la serialización y deserialización eficiente de mensajes estructurados entre cliente y servidor.
protobufjs-cli	1.1.3	Interfaz para línea de comandos de `protobufjs`. Facilita la generación de archivos JavaScript y TypeScript de esquemas compatibles de Protocol Buffers.
ts-node	10.8.1	Ejecuta archivos TypeScript directamente en Node.js. Se usa para desarrollo rápido y ejecución sin compilación previa, empleado para agilizar la evaluación en entornos de pruebas.
typescript	5.8.3	Lenguaje de programación que extiende JavaScript. Aporta robustez al proyecto, facilita el mantenimiento del código, y previene errores comunes en tiempo de compilación.

En la fase de conexión inicial (cuando un dispositivo establece una conexión TCP con el sistema), se identifica al dispositivo por medio de una huella digital generada al concatenar la dirección IP del cliente, una marca de tiempo tomada al momento de la solicitud y su respectivo socket de referencia. Esta cadena compuesta se somete a una función hash proporcionada por el módulo crypto de Node.js, cuya salida es un identificador único e irrepetible, útil para identificar a los distintos clientes.

Por ejemplo, si un cliente accede desde una red doméstica con la dirección IP 2.71.82.81, el servidor añade a esta dirección una marca de tiempo generada al recibir la petición y otros elementos del entorno de conexión, obteniendo una cadena de datos con la estructura "2.71.82.81–1070720251–socketRef", que es procesada por el algoritmo de hashing para producir un ID con el formato "48656c6c6f21...". Los primeros 8 elementos de este hash son asignados al cliente y se convierte en su identificador persistente a lo largo de esa sesión, transmitiéndose al cliente como parte del mensaje de bienvenida.

Para supervisar el estado de cada conexión, un mecanismo de latidos (*heartbeat*) rastrea la capacidad de respuesta de los usuarios. Con este mecanismo, el servidor espera un mensaje de latido de cada cliente en intervalos de 60 segundos (`heartbeat_interval`). Al recibir cada latido o cualquier otra operación realizada por el cliente, el servidor actualiza internamente el registro de actividad del cliente, garantizando que el sistema tenga siempre un estado preciso de la sesión. Si se acumulan tres intervalos consecutivos sin recibir latidos desde un cliente (`missedHeartbeats`), el sistema lo marca como ausente y emite una notificación al resto de los usuarios de la sala, indicándoles la posible desconexión, sin limitar sus funciones dentro de la sesión. Esta operación no implica la eliminación definitiva del cliente, sino que abre una ventana para su reconexión. Si dentro de los próximos 320 segundos posteriores a esta notificación no se recibe una actualización por parte del cliente, el sistema considera que la desconexión es definitiva, eliminando su registro de actividad, liberando los recursos asociados al cliente y se actualiza la composición de la sala.

4.2.2. Gestión de usuarios

El módulo de gestión de usuarios constituye el bloque encargado de representar, supervisar y mantener actualizado el estado de cada cliente conectado en tiempo real, garantizando la coherencia del entorno colaborativo y la integridad de las comunicaciones. Cada usuario se abstrae mediante una estructura en memoria conforme a la interfaz `IPlayer` ilustrada en la Tabla 4.2.

La estructura `IPlayer` incluye un campo `id` que almacena el identificador único del cliente, generado por el servidor a partir de la función hash descrita en la sección 4.2.1. El campo `conexion` registra la referencia del socket asociado a la conexión TCP del cliente, permitiendo intercambiar datos entre el servidor y el cliente sin otros intermediarios. El atributo `roomId` almacena el identificador de la sala colaborativa en la que se encuentra activo el usuario, mientras que el atributo `lastSeen` y `lastActivity` registran las marcas temporales más recientes del último latido recibido y de la última interacción significativa respectivamente, ambos utilizados para el cálculo de latencia y detección de inactividad.

Tabla 4.2 Atributos y métodos públicos de la interfaz `IPlayer`

Atributo	Descripción
`id`	Identificador único del jugador.
`conexion`	Socket activo que representa la conexión del jugador.
`roomId`	Identificador de la sala en la que se encuentra el jugador.
`lastSeen`	Marca temporal de la última vez que el jugador fue detectado.
`isConnected`	Estado de conexión actual del jugador.
`lastActivity`	Fecha de la última actividad registrada.
`heartbeatInterval`	Intervalo de latidos para verificar la conexión.
`missedHeartbeats`	Número de latidos perdidos consecutivos.
`buffer`	Cadena de datos en espera de procesamiento.
Método	**Descripción**
`cleanup()`	Libera recursos y elimina referencias del jugador.
`serializer()`	Devuelve el manejador de serialización para el jugador.

El campo `isConnected` actúa como bandera lógica para determinar si una conexión está activa y operativa, sincronizándose con el atributo `missedHeartbeats` quien lleva el conteo acumulado de latidos no recibidos, lo que permite enviar respuestas reactivas a la pérdida de conexión y activar rutinas de recuperación. Por su parte, el `heartbeatInterval` guarda una referencia al temporizador asignado a cada cliente con la finalidad de registrar la periodicidad de los latidos, mientras que `buffer` actúa como espacio temporal para la serialización y deserialización de los datos.

También se definen dos métodos fundamentales como `cleanup()`, encargado de liberar los datos asociados a un cliente cuando la conexión se cierra o expira, y `serializer()`, que

expone la lógica de serialización específica empleada por el cliente, permitiendo interpretar adecuadamente los datos entrantes.

4.2.3. Matchmaking

El bloque de emparejamiento o *matchmaking* define y operacionaliza el concepto de sala colaborativa como un elemento funcional del sistema. Una sala colaborativa se define como un espacio lógico acotado donde un grupo específico de usuarios comparte en tiempo real objetos, eventos y estados sincronizados. El sistema es capaz de gestionar hasta n salas colaborativas, cada una con una capacidad máxima de m usuarios concurrentes en espacios lógicos aislados.

Matchmaking gestiona de forma automatizada la asignación de nuevos clientes a salas colaborativas existentes o, en su defecto, la creación de nuevas salas de ser necesarias. Esta asignación se rige por un esquema secuencial de registro en memoria FIFO (*First-In, First-Out*), donde las referencias de las conexiones se insertan ordenadamente en un arreglo dinámico que representa a los usuarios de una sala. El primer cliente en ingresar ocupa la posición 0, y los siguientes clientes se añaden de manera secuencial conforme a su orden de llegada. Cuando se requiere extraer información de los participantes, el sistema accede a las posiciones del arreglo conforme fueron insertadas.

El servidor también almacena todas las salas activas, cada una de ellas con su respectivo identificador y referencias a clientes, que consulta para verificar la disponibilidad de espacio en alguna de ellas. El criterio principal de emparejamiento se basa en la disponibilidad de las salas ya creadas: si una sala no ha alcanzado el número máximo m de usuarios permitido, se vincula al nuevo jugador a dicha sala y se procede a activar los mecanismos colaborativos correspondientes. En caso contrario, se crea una nueva sala colaborativa y se coloca al cliente en espera como su primer miembro, manteniendola inactiva hasta conseguir el número máximo m de usuarios.

El número de usuarios m por sala es un parámetro configurable previo a la inicialización del servidor. Este parámetro permite modelar escenarios diversos: desde experiencias uno a uno, hasta dinámicas multijugador. Cada sala colaborativa actúa como una unidad encapsulada del entorno colaborativo global. Dentro de su lógica se mantienen referencias explícitas a los dispositivos conectados y a los objetos virtuales generados, como modelos manipulables, trazos gráficos o registros de interacción.

4.2.4. Manejador de serialización

Este bloque garantiza la comunicación entre clientes heterogéneos. El manejador adopta un mecanismo similar a [80] capaz de alternar dinámicamente entre distintos formatos de serialización en lugar de la capa de transporte, sin alterar la semántica del entorno

colaborativo. Esta característica se consigue con un diseño basado en interfaces, donde cada modelo de serialización, como JSON, MessagePack o Protobuf, se encapsula en una clase común denominada `SerializationHandler`. La definición de operaciones esenciales como la codificación y decodificación automatizada se consigue con el uso de los identificadores MIME (*Multipurpose Internet Mail Extensions*) correspondientes, que son etiquetas estandarizadas que permiten al servidor reconocer y tratar adecuadamente cada mensaje entrante o saliente según los primeros caracteres recibidos en su formato.

El sistema, por practicidad, permite que la elección del serializador no sea una configuración global impuesta por el mismo servidor, sino una decisión contextual determinada por los clientes. Esta decisión puede interpretarse desde el mensaje inicial enviado por el cliente o establecerse explícitamente como parte del protocolo de conexión. Esta solución garantiza la interoperabilidad entre dispositivos que, por compatibilidad, recursos o rendimiento computacional, requieren emplear distintos formatos. Por ejemplo, alguna implementación desarrollada con editores actuales de Unity Engine podrían hacer uso de Protobuf para aprovechar su bajo costo computacional y capacidad de transmisión más eficiente [119], mientras que aplicaciones Web o clientes no compatibles pueden continuar empleando el formato de serialización JSON. La decisión de desacoplar completamente el formato de serialización del núcleo lógico del servidor fortalece además su capacidad de evolución futura ante su posible incorporación de nuevos formatos, ya sea por necesidad técnica, avance tecnológico o integración con plataformas externas.

Tabla 4.3 Atributos y métodos públicos de la clase `Room`

Atributo	Descripción
`roomId`	Identificador único de la sala.
`players`	Lista de usuarios conectados, representados mediante objetos del tipo `IPlayer`.
`objects`	Estructura que contiene los elementos posicionados en el espacio, definidos por coordenadas del tipo `Vector3`.
`drawings`	Trazos realizados por los participantes, también representados por el tipo de dato `Vector3`.
`lastActivity`	Marca temporal de la última actividad registrada en la sala.
Método	**Descripción**
`broadcast(data: String)`	Envía datos a todos los usuarios conectados.
`addPlayer(player: IPlayer)`	Añade un nuevo participante a la sala.
`removePlayer()`	Remueve un usuario identificado previamente de la sala.
`playerReconnected()`	Reestablece el estado de un usuario tras su reconexión.
`playerDisconnected()`	Marca a un participante como desconectado.
`isEmpty()`	Comprueba si no hay usuarios activos en la sala.
`getActivePlayers()`	Devuelve el conjunto de usuarios actualmente conectados.

4.2.5. Bloque de difusión

El bloque de difusión o *broadcasting* se encarga de propagar instantáneamente los mensajes recibidos hacia el resto de los participantes que comparten una misma sesión. Su función central, `broadcast()`, permite emitir mensajes a todos los integrantes activos de una sala determinada, excluyendo al emisor inicial para reducir el tráfico en la red local.

El método `broadcast()` forma parte de una abstracción lógica denominada `Room` (ver Tabla 4.3), una entidad que describe la lógica de cada sesión colaborativa. Al invocarse el método de difusión, el sistema transmite a cada usuario asociado a la misma sala correspondiente la estructura y secuencia presentada en la sección 4.2.3.

La difusión de eventos habilita el núcleo de la experiencia colaborativa: cada acción significativa, como iniciar un trazo de dibujo, actualizar una posición o señalar una entidad compartida, es replicada por los demás participantes, reforzando la sensación de interacción simultánea. Con cada evento de transmisión, la marca de tiempo de actividad de la sala es actualizada, lo cual alimenta el proceso de detección de inactividad. Si durante un periodo prolongado no se registran eventos transmitidos ni interacciones entre participantes, el sistema detecta la inactividad de la sala y también procede a desasignar los recursos vinculados, optimizando el consumo de memoria y reduciendo la carga del sistema.

4.3. Diseño de la arquitectura descentralizada

La arquitectura descentralizada fue diseñada con el objetivo de facilitar la interacción punto a punto entre nodos distribuidos, prescindiendo de una infraestructura centralizada. La solución adoptó un patrón de mensajería pub/sub (publicador/suscriptor) similar al sistema de Pereira *et al.* [101], pero implementado sobre la pila tecnológica de NetMQ, una adaptación completamente nativa en C-Sharp de ZeroMQ [120], donde los emisores (publicadores) generan mensajes sin requerir conocimiento explícito de los receptores (suscriptores).

La comunicación se organiza mediante tópicos (*topics*), estructurando el flujo de datos en canales temáticos diferenciados según su función. Cada *topic*, se asigna a un canal específico de comunicación, por ejemplo, un *topic* para compartir trazos de dibujo y anclajes espaciales, un segundo *topic* para compartir posición u orientación de los clientes, o un tercer *topic* para presentarse ante el resto de nodos, asegurando que todos los nodos suscritos a esos tópicos reciban las actualizaciones pertinentes en tiempo real. Este enfoque también permite la incorporación dinámica de nuevos participantes y se elimina el concepto de sala colaborativa: cualquier nodo que se conecte y se suscriba al tópico correspondiente comenzará a recibir información sin necesidad de reconfigurar el resto de la red.

4.3.1. Mecanismo de descubrimiento de nodos

Uno de los desafíos iniciales a los que se enfrenta el diseño de esta arquitectura consiste en la detección de los dispositivos activos que comparten la misma red. Al eliminar el uso de un servidor central, cada nodo pierde el acceso a una fuente común de información que, en las arquitecturas centralizadas tradicionales, facilita la conexión entre los participantes. En ausencia de este componente intermediario, se requiere un método alternativo que permita a los dispositivos detectar de forma autónoma a sus pares activos.

Para resolver este problema, se desarrolló el componente `MulticastDiscovery`, responsable de realizar el descubrimiento de nodos basado en mensajes UDP como mecanismo equivalente al empleado por Brown *et al.* [80]. A través de este mecanismo, los nodos de la red envían anuncios a la dirección de multidifusión 224.0.0.1, una dirección IP reservada para enviar paquetes de datos a todos los miembros de un grupo, en lugar de su envío a cada receptor. Este canal funciona como espacio compartido común donde los nodos anuncian su presencia en la red. Al iniciar la aplicación, cada nodo transmite periódicamente un mensaje que incluye su dirección IP, el puerto de servicio y los tópicos donde publica. La estructura de este mensaje se describe en la Tabla 4.4:

Tabla 4.4 Estructura del mensaje de descubrimiento enviado por cada nodo

Campo	Descripción
`ip`	Dirección IP del dispositivo que expone servicios de publicación en la red local.
`nodeId`	Identificador único del nodo, utilizado para rastrear su disponibilidad y validar su reconocimiento por otros nodos.
`topics`	Lista de tópicos en los que el nodo publica información, correspondiente a su perspectiva del entorno.

4.3.2. Gestión de sockets y conexiones

Para asegurar la consistencia de la red, el módulo `MulticastDiscovery` implementa un sistema de control de vida de los nodos. Cada anuncio recibido por UDP actualiza el registro local de cada dispositivo identificado en la red, posteriormente actualiza una marca de tiempo asociada al nodo que realiza la transmisión en su propia copia de participantes activos. Si un nodo deja de enviar anuncios dentro de un intervalo preestablecido (60 segundos), se considera inactivo y se elimina de la lista local, lo que desencadena la desconexión de sus correspondientes suscripciones. Este mecanismo asegura que los dispositivos no intenten comunicarse con nodos que ya no se encuentran disponibles, reduciendo así errores y sobre carga a la red innecesaria. El mecanismo contempla los siguientes tres tipos de desconexiones:

- Por inactividad: Se gestiona mediante un sistema de tiempo de espera preestablecido. Si un nodo continúa enviando mensajes dentro del intervalo, se actualiza su marca de tiempo. En caso contrario, se elimina de la lista de nodos disponibles. Si posteriormente se reconecta, sus mensajes de anuncio permiten su reintegración como un nuevo nodo descubierto.
- Por el usuario: Ocurre cuando el participante decide abandonar la sesión.
- Forzada: Se produce por fallos críticos en la aplicación, como errores lógicos o sobrecarga de recursos.

Por conveniencia, el sistema define un tópico principal denominado "ARDrawing", al cual todos los nodos están programados para suscribirse una vez inician la conexión. Un segundo módulo identificado como `CommunicationManager` proporciona métodos para orquestar la publicación de mensajes en este canal empleando sockets TCP, asegurando que los paquetes de datos que encapsulan los trazos y anclajes espaciales en estructuras serializadas lleguen integramente a cada nodo. Cada mensaje incluye el identificador del anclaje, la secuencia de puntos del trazo, atributos visuales y coordenadas de referencia, como se empleó en la arquitectura centralizada en la Sección 4.2.4.

4.3.3. Flujo de operación del sistema

El comportamiento de la arquitectura descentralizada se comprende mejor examinando el ciclo de transmisión de cada mensaje, desde la construcción hasta la replicación en todos los dispositivos participantes. Este proceso, ejemplificado en la Figura 4.3, se basa en la interacción coordinada de los tres módulos principales: `DrawManager` (encargado de procesar eventos de interacción), `CommunicationManager` (encargado de orquestar la transmisión de los datos) y `MulticastDiscovery` (encargado de validar los nodos activos).

Cuando un usuario realiza un trazo, `DrawManager` construye y registra la nube de puntos generada por la interacción. Posteriormente, su sistema coordenado es ajustado para garantizar su persistencia dentro del entorno. Una vez construido el mensaje, se serializa en conjunto a su geometría, sus atributos visuales y su referencia espacial correspondiente.

El módulo `CommunicationManager` publica en su propio tópico de dibujo este paquete serializado mediante su socket de publicación, cuyos nodos ya se encuentran suscritos y actualizados por el módulo `MulticastDiscovery`. Este componente gestiona la detección de los dispositivos activos mediante mensajes periódicos de multidifusión a través de la dirección estándar 224.0.0.1 y el puerto 2718, configuración establecida para facilitar la implementación en el entorno de pruebas, con lo que se asegura distribuir la información únicamente a los participantes presentes en la misma sesión.

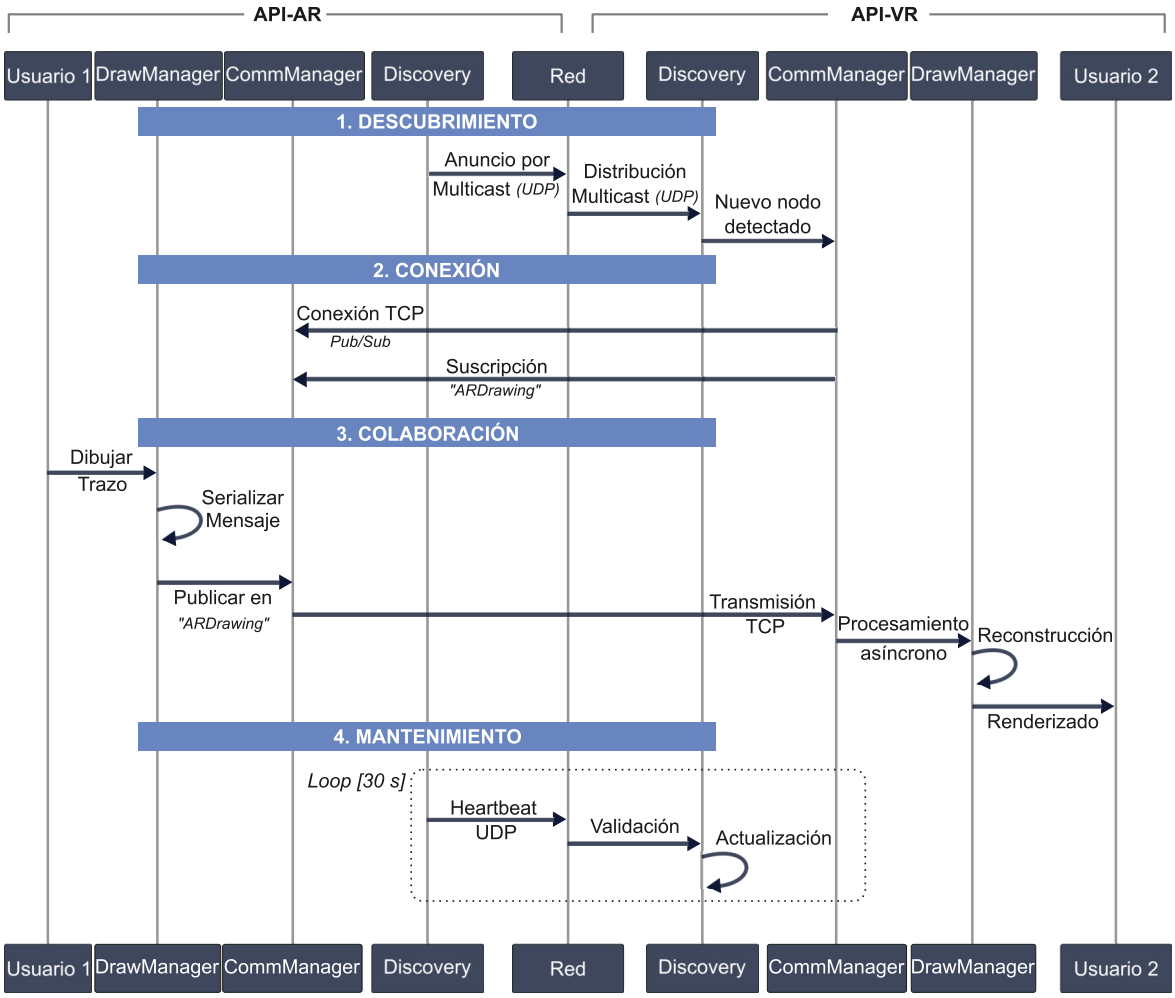


Figura 4.3: Diagrama de composición de la arquitectura descentralizada SRV–D.

En el nodo suscriptor, el módulo `CommunicationManager` opera de forma asíncrona, procesando los mensajes recibidos a través de un hilo dedicado para su visualización. `ARDrawManager` procesa estos mensajes, reconstruyendo el trazo en la escena del nodo suscriptor. Como resultado, el usuario remoto observa la aparición progresiva del trazo realizado, permitiendo una colaboración continua entre todos los participantes.

4.4. Diseño del entorno interactivo API–AR

La plataforma API–AR, ilustrada en la subfigura (C) de la Figura 4.1, constituye el entorno desde el cual se analiza el desempeño de ambas arquitecturas propuestas desde una perspectiva aumentada. Desarrollada en Unity Engine 2022.3.55f1, motor de aplicaciones compatible con Google ARCore, esta integración incluye extensiones que posibilitan capacidades de realidad aumentada en dispositivos Android. Gracias a ello, múltiples dis-

positivos móviles certificados por el proveedor [43] pueden participar simultáneamente en un entorno de anotación espacial, donde se crean y editan objetos gráficos tridimensionales superpuestos en el mundo físico.

El funcionamiento de API–AR se basa en la captación constante del mundo físico, apoyándose en el seguimiento espacial, la detección de planos, la estimación de luz y reconstrucción parcial del ambiente mediante cámaras e información inercial. Sobre esta base sensorial, la aplicación superpone los elementos virtuales en el espacio real, permitiendo interacciones naturales a través de la pantalla táctil. Su diseño cumple un doble propósito: por un lado, actúa como medio para la generación de trazos tridimensionales anclados al entorno, por otro, se establece como una plataforma de evaluación sistemática para ambas arquitecturas propuestas, permitiendo medir métricas como la latencia de propagación de los mensajes, la consistencia espacial entre los dispositivos y la robustez frente a fallos o desconexiones temporales.

4.4.1. Diseño funcional

El diseño funcional de API–AR se organiza en cuatro subsistemas coordinados que permiten su análisis y extensión por separado:

1. **Gestión de entrada gestual**: Este módulo interpreta las interacciones realizadas a través de la pantalla táctil del teléfono móvil. Con el uso del sistema `EnhancedTouch` de Unity Engine, se capturan eventos como toques, deslizamientos y arrastres, los cuales son procesados para generar trazos virtuales sobre el espacio aumentado. La interpretación de gestos permite reconocer secuencias espaciales y temporales que definen acciones complejas, como iniciar, continuar o finalizar un dibujo.

2. **Subsistema de generación de trazos**: Partiendo de las interacciones capturadas en la entrada gestual, este subsistema crea líneas tridimensionales (`ARLines`), cada una con un color, textura y tamaño que son renderizados dinámicamente en la escena de RA. Cada línea se construye con un conjunto finito y espaciado de puntos espaciales, interpolados conforme su orden de creación, para su visualización como un elemento continuo e ininterrumpido, incluso si el dispositivo cambia de ubicación o reinicia su percepción del entorno.

3. **Sincronización espacial mediante anclajes**: Un anclaje (*anchor*) en el contexto de la RA es un punto de interés en el espacio que se emplea como referencia espacial para posicionar objetos digitales [121]. Para ello, se hace uso de la odometría visual-inercial de Google ARCore para detectar planos o superficies estables del entorno y establecer sobre ellos anclajes persistentes. Estos anclajes se replican mediante la transmisión de sus metadatos asociados, consiguiendo una representación coherente, incluso si la percepción espacial entre dispositivos difieren ligeramente.

4. **Capa de comunicación y serialización**: Este subsistema se encarga de empaquetar la información generada (trazos, anclajes, eventos de sesión) en mensajes estandarizados que son enviadas a través de la red. La aplicación admite múltiples esquemas de codificación, como JSON, MessagePack y Protobuf, seleccionables dinámicamente según el escenario de evaluación.

Respecto al diseño visual e interactivo, la interfaz API–AR se organiza en los siguientes dos componentes complementarios:

- **Componente tridimensional**: Esta capa constituye el núcleo de la experiencia aumentada. Es aquí donde se proyectan, en tiempo real, los trazos dibujados, los anclajes establecidos y las anotaciones compartidas.
- **Componente bidimensional de control (UI canvas)**: Este componente superpone elementos gráficos tradicionales como botones, menús, imágenes, y controles visuales sobre el componente tridimensional. Su propósito es permitir el acceso inmediato a las funciones de edición y configuración de los trazos tridimensionales sin interferir con el entorno aumentado.

4.4.2. Sincronización espacial mediante anclajes persistentes

La persistencia espacial de los objetos virtuales, entendida como la capacidad de los objetos de mantenerse fijos en posiciones coherentes, se consigue empleando anclas (*anchors*), que son puntos de referencia tridimensionales sobre los cuales se posicionan los elementos aumentados. Evitando el uso de servicios propietarios como Cloud Anchors de Google (que requieren acceso a internet, permisos de localización y licencias de uso), se diseñó un protocolo personalizado de sincronización usando un sistema de coordenadas relativas.

La estrategia de sincronización implementada parte de la siguiente simplificación: todos los dispositivos inician su sesión colaborativa en una posición física aproximadamente coincidente, ilustrado en la subfigura (A) de la Figura 4.1. A partir de esta suposición, en cada dispositivo se define su propio sistema de referenciación espacial, cuyo origen local se define como $(x, y, z) = (0, 0, 0)$, siendo este el lugar en el entorno real donde se inicia la aplicación. Si bien esta aproximación presenta limitaciones en entornos muy amplios o dispersos, resulta suficiente para sesiones controladas con usuarios colocalizados.

Cada *anchor* identificado tiene asociado una entidad lógica compuesta por:

1. Un identificador único (UUID) generado localmente,

2. Una posición tridimensional relativa al dispositivo que procesa el *anchor*,
3. Y una orientación espacial.

Cuando el *anchor* es creado, sus metadatos se replican en el resto de dispositivos mediante un mensaje estructurado haciendo uso de alguna de las arquitecturas de red desarrolladas. Empleando los identificadores persistentes de cada *anchor*, los clientes reconstruyen los metadatos recibidos desde su marco de referencia local, interpretándolo como una posición relativa al origen con el que iniciaron su sesión.

Este modelo de sincronización tiene dos ventajas principales. La primera ventaja es que no se requiere el uso de mapas espaciales compartidos o reconstrucciones simultáneas del entorno como el de Han *et al.* [90], lo que simplifica el procesamiento computacional y reduce el tráfico de datos. La segunda ventaja es que permite una evaluación controlada de las capacidades en ambas arquitecturas de red. Además, el sistema extiende su alcance y resuelve otros desafíos como oclusiones temporales, cambios de iluminación o la captura y procesamiento de datos visuales a través de Google ARCore. Cuando el sistema de percepción del dispositivo pierde momentáneamente el seguimiento espacial, se recurre a anclajes previamente establecidos por el dispositivo para estimar su transformación espacial con respecto a su estado anterior.

4.4.3. Generación y transmisión de trazos espaciales

La generación de contenido gráfico se basa en una gramática gestual diseñada específicamente para interpretar las interacciones que suceden sobre la pantalla táctil del dispositivo. Esta gramática permite detectar eventos discretos (como toques iniciales) y movimientos continuos (arrastres o deslizamientos), los cuales son proyectados como puntos tridimensionales. Cada interacción con la pantalla táctil da lugar a la creación de un `ARLine`, un contenedor de líneas tridimensionales sobre el entorno aumentado (ver Figura 4.4). Esta línea es referenciada al *anchor* más cercano dentro del entorno físico, consiguiendo así mantener una posición fija y coherente, incluso si el dispositivo que lo creó se desplaza o pierde su seguimiento temporal.

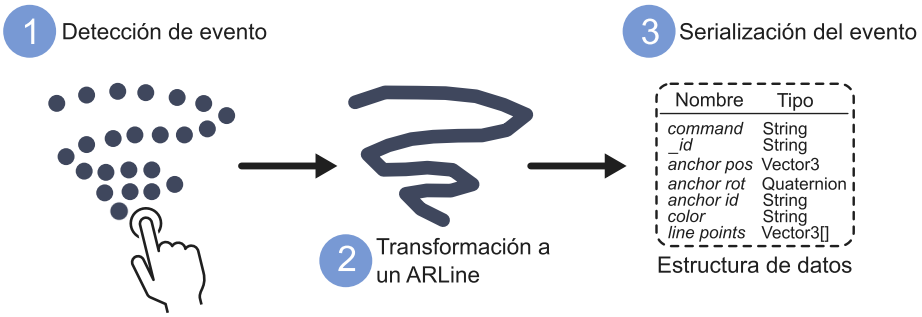


Figura 4.4: Proceso de generación de trazos espaciales.

Durante la interacción, los puntos 2D capturados en pantalla son proyectados 60 centímetros sobre el espacio tridimensional mediante un rayo virtual que parte de la cámara de RA hacia el entorno físico. Los puntos capturados se interpolan linealmente para producir el efecto visual de un trazo continuo y sus propiedades como el color, grosor y opacidad se gestionan a través de `LineSettings`, una estructura configurable a través del componente bidimensional de control (UI canvas) descrito en la Sección 4.4.1.

Finalizado un trazo (por ejemplo, al levantar el dedo de la pantalla), inicia la operación de serialización. En esta etapa, los metadatos asociados al trazo se encapsulan bajo la estructura denominada `Drawings`, conformada por los siguientes atributos:

- Identificador único del usuario que generó el trazo.
- Identificador de la sala colaborativa en la que se encuentra.
- Identificador del anclaje al que está vinculado el trazo.
- Posición espacial del anclaje en coordenadas relativas al sistema local.
- Lista de puntos que definen la estructura espacial del trazo.
- Propiedades visuales: color y grosor del trazo.

Esta estructura se serializa mediante el esquema JSON, MessagePack o Protobuf. La elección del formato solo responde a criterios evaluativos.

4.4.4. Recepción, reconstrucción y animación de trazos remotos

Cuando se recibe un mensaje de dibujo generado por otro participante de la sesión colaborativa, se inicia la operación de reconstrucción local. En este procedimiento se extrae del mensaje recibido el identificador único del *anchor*, su posición relativa donde fue creado, el conjunto de puntos espaciales que componen al trazo, así como sus metadatos visuales. A continuación, se construye un nuevo *anchor* con las mismas propiedades compartidas a través del mensaje, pero para su visualización se ejecuta una animación progresiva que distribuye el proceso de reconstrucción gráfica a lo largo de varios cuadros de renderizado (*frames*), dando la ilusión de ser reconstruido de manera natural y no instantáneamente.

4.5. Diseño del entorno interactivo API–VR

Además de su implementación en dispositivos móviles, el sistema global fue adaptado para operar en entornos virtuales. Esta versión, denominada API–VR, se diseñó con el

objetivo de conservar los principios funcionales de la versión en RA. Mientras que la versión en RA funciona en dispositivos móviles compatibles con ARCore, API–VR está pensada para plataformas que soportan interfaces XR, como OpenXR u Oculus SDK, por lo que requiere el uso de equipos con visores y controladores para RV.

Esta aplicación también se realizó empleando Unity Engine 2022.3.55f1, pero utilizando el sistema XR Interaction Toolkit, una librería de Unity diseñada para facilitar la creación de entornos interactivos en RV. Este toolkit consigue detectar entradas desde controladores, manejar colisiones espaciales, definir zonas de interacción y vincular gestos físicos con acciones programadas.

API–VR también permite visualizar trazos construidos por otros participantes que usan dispositivos móviles o distintos controladores de RV, pero bajo un ambiente más controlado respecto a API–AR, ya que se eliminan las condiciones físicas como iluminación, sombras y oclusiones.

4.5.1. Lógica de interacción

La interacción en RV es radicalmente distinta a la de los dispositivos móviles basados en pantalla táctil. Contrario a una interacción multitoque (pulsaciones, deslizamientos o pellizcos), el sistema en RV emplea un par de periféricos con seguimiento espacial, comúnmente conocidos como mandos o controladores de RV. Estos controladores permiten capturar simultáneamente la posición x, y, z, así como la orientación (*roll*, *pitch* y *yaw*), habilitando una interacción más exacta y precisa sobre los objetos virtuales.

Por ello, la mecánica de interacción fue rediseñada de tal manera que la manipulación de los objetos virtuales fuese mapeada a través de los botones físicos del controlador. Cada trazo se inicia al presionar el gatillo o "trigger" del controlador derecho (*right controller*), mientras su visualización es representada mediante el componente `TrailRenderer`, un componente nativo de Unity Engine con el que se generan estelas o trazos visuales.

`TrailRenderer` no solo visualiza la trayectoria generada, también almacena internamente los puntos de la trayectoria que se ha seguido mientras el gatillo se mantiene presionado. Una vez se deja de pulsar, se finaliza el trazo y se procede a su envío conforme a la estructura empleada en API–AR. Es importante destacar que la estela generada se separa inmediatamente del controlador tras completar el trazo, estableciendo su propia posición fija en el espacio virtual, de manera equivalente a las anclas generadas en API–AR.

4.5.2. Lógica de sincronización espacial

API–AR y API–VR comparten un sistema común de coordenadas basado en anclas espaciales persistentes. Mientras que en el entorno de RA un ancla se asocia a una posición física, en RV estos anclajes se definen en un espacio virtual que emula la distribución del espacio físico. La clave de esta coherencia espacial se consigue a través de una reinterpretación de los anclajes.

Cuando un usuario de la interfaz API–VR procesa la información de un trazo generado en RA, la geometría del trazo se reconstruye en su ubicación relativa incorporando, además, una compensación basada en un desfase de altura de aproximadamente 1.15 metros. Este ajuste minimiza la diferencia entre el origen de ambos espacios coordenados, uno centrado en la posición inicial de la cámara del dispositivo móvil, y el otro en la posición inicial de las gafas de RV.

Este desplazamiento permite que un trazo realizado sobre una superficie física se proyecte coherentemente en el espacio virtual, evitando desalineaciones que comprometan la percepción compartida.

Capítulo 5

Pruebas y resultados

En este capítulo se describe el protocolo de pruebas realizado, los recursos disponibles para evaluar ambas arquitecturas propuestas y se presentan los resultados obtenidos en el proceso de evaluación. Finalmente, se concluye con un análisis de rendimiento, así como los criterios de satisfacción de los objetivos inicialmente propuestos.

5.1. Metodología y validación experimental

API–AR y API–VR no representan únicamente una interfaz de usuario, sino que fungen el rol de intermediarios para evaluar ambas arquitecturas de red desarrolladas. Ambas aplicaciones incluyen las herramientas necesarias para dirigir un análisis experimental a través de dos dimensiones: el rendimiento del sistema (enfocado en comparar la latencia de la comunicación) y la consistencia colaborativa (enfocada en contrastar la fidelidad de replicación). Para ello, todas las pruebas realizadas se implementaron en un entorno interior controlado, con iluminación estable y obstrucciones físicas mínimas, con la finalidad de garantizar un seguimiento espacial consistente y la reproducibilidad experimental en escenarios similares.

El conjunto de pruebas se realizó entre dos usuarios: un teléfono inteligente (*smartphone*) compatible con ARCore y la aplicación API–AR y un sistema de realidad virtual basado en PC equipado con un controlador de 6 grados de libertad, compatible con la aplicación API–VR. Ambos dispositivos compartieron un espacio físico de aproximadamente 3×3×3 metros, manteniendo una colocalización espacial durante todo el experimento.

Tabla 5.1 Especificaciones de los dispositivos utilizados en el banco de pruebas

Dispositivo	Plataforma	Entrada	Descripción
Smartphone (Dispositivo A)	Android 11	Pantalla táctil, seguimiento por cámara	Usuario API–AR ejecutándose en Motorola One Hyper.
Visores de RV (Dispositivo B)	Windows 11	Controlador manual de 6DoF	Usuario API–VR ejecutándose en Oculus Rift y Alienware Aurora R7.
PC	macOS 15.5	N/A	Instancia del servidor SRV–C en Mac Studio M2 Max.

Al inicio de cada prueba, los dispositivos se colocaron uno junto al otro en un origen espacial común, con el dispositivo de RA adyacente al visor de RV. Esto aseguró que ambos usuarios experimentaran el entorno colaborativo desde un punto de partida equivalente, facilitando un anclaje espacial consistente y la alineación de anotaciones entre las dos plataformas.

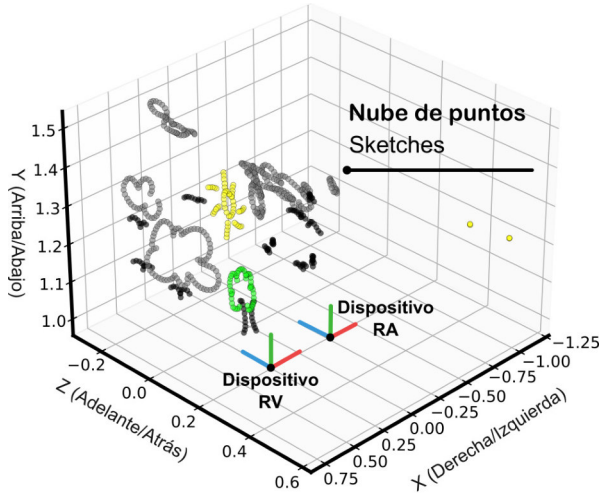


Figura 5.1: Mapa de distribución espacial (distancias en metros). La nube de puntos indica un ejemplo de las localizaciones de muestreo generadas (*sketches*) por ambos usuarios.

La conectividad de red se estableció mediante una LAN usando un *router* ASUS RT-AC5300 operando en la banda de 5 GHz, a una tasa de datos de hasta 1734 Mbps de acuerdo con las especificaciones proporcionadas por el fabricante para la interfaz 802.11ac. El *router* emitía múltiples SSIDs, sin embargo, todos los dispositivos de prueba se conectaron exclusivamente a un SSID dedicado. No hubo dispositivos adicionales conectados a la red específica de pruebas durante las mediciones.

La transmisión de los paquetes se manejó exclusivamente mediante sockets TCP persistentes. Para el caso de la arquitectura centralizada, se requirió una instancia del servidor SRV–C en una máquina PC (ver Tabla 5.1), mientras que, para la arquitectura descentralizada, la comunicación se realizó entre los dispositivos, sin depender de infraestructuras externas o en la nube.

Los datos espaciales brutos capturados de la entrada del usuario (nube de puntos) se procesaron utilizando la función Vector3.Lerp de Unity Engine para realizar una interpolación lineal entre los datos muestreados en cada evento. Este paso generó un trazo continuo y uniforme, mejorando la suavidad y consistencia de la representación visual de cada trazo, transmitiéndose como un paquete de datos continuo para emular el dinamismo

de la entrada del usuario en tiempo real.

Bajo esta configuración, los usuarios de RA y RV crearon colaborativamente anotaciones 3D que permanecieron ancladas en el espacio (*sketches*), como se visualiza en la Figura 5.1, empleando ambas arquitecturas de manera separada para evaluar su capacidad de respuesta bajo condiciones realistas de red y seguimiento.

5.1.1. Configuración de los trazos

Para explorar un amplio rango de escenarios colaborativos, se definieron seis figuras geométricas como plantillas de trazado, ilustradas en la Figura 5.2. Estas plantillas se superpusieron visualmente en cada participante, sirviendo como guía dentro de un área de dibujo de aproximadamente 25×25 centímetros. A cada participante se le indicó que replicara las figuras en el espacio 3D trazando secuencialmente los vértices de la plantilla en la dirección indicada, adaptando el protocolo presentado en [122].

La evaluación se realizó de forma bidireccional. Primero, el dispositivo de RA realizaba las tareas de dibujo mientras que el dispositivo de RV realizaba la tarea de replicación. Después, los roles se invertían, siguiendo esta metodología en ambas arquitecturas. Cada figura se replicó 50 veces en ambos dispositivos, resultando en 300 pruebas por dirección de transmisión y un total de 600 eventos anotados para cada tipo de arquitectura evaluada.

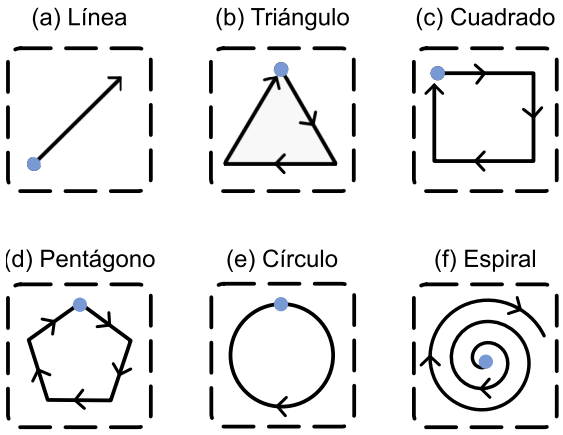


Figura 5.2: Seis figuras geométricas utilizadas como plantillas de trazado para probar ambas arquitecturas desarrolladas. Las plantillas incluyen: (a) Línea, (b) Triángulo, (c) Cuadrado, (d) Pentágono, (e) Círculo, y (f) Espiral. Los participantes trazan estas figuras en el espacio 3D, siguiendo la dirección indicada por las flechas, para replicarlas con precisión. Los puntos azules representan el punto de inicio de cada trazo.

Si bien la plantilla visual estandarizó la ruta y el tamaño del trazo, la velocidad de dibujo estuvo sujeta a variaciones naturales entre los participantes. Aunque se hicieron esfuerzos para fomentar un ritmo moderado y constante en el dibujo de los trazos, no

se impuso como una variable controlada. Dado que las principales métricas de interés miden el tiempo de procesamiento del sistema por trazo (después de su finalización) y no el rendimiento en tiempo real del envío punto por punto, se considera que el impacto de la velocidad de dibujo en los resultados es mínimo. A lo largo de todas las pruebas, ambos dispositivos registraron marcas de tiempo detalladas para cada trazo, incluyendo: el tiempo de transmisión del mensaje al receptor, el tiempo de recepción de la respuesta de eco del servidor (para la arquitectura centralizada) o del dispositivo emparejado (para la arquitectura descentralizada) y el tiempo de finalización de la reconstrucción del trazo.

5.1.2. Métricas de evaluación

Para abordar la desincronización de los relojes entre dispositivos, se implementó un protocolo personalizado de eco de mensajes. Las mediciones de latencia se realizaron en rondas de transmisión alternas, donde solo un dispositivo actuaba como emisor por prueba.

En la arquitectura centralizada, al recibir un mensaje, el servidor SRV–C lo retransmitía inmediatamente a todos los participantes, incluido el emisor original. El emisor registraba tanto las marcas de tiempo de transmisión como de recepción del eco usando su propio reloj local, mientras que el receptor registraba de forma independiente las marcas de tiempo de recepción y reconstrucción dentro de su propio dominio de tiempo. Por el contrario, en la arquitectura descentralizada, cada publicador transmitía el mensaje completo hacia un tópico especifico, registrando los instantes de tiempo de transmisión como de recepción del eco enviado por el suscriptor, ambos usando su propio reloj local. De esta manera, el emisor mide el tiempo de respuesta en la red, mientras que el receptor mide el tiempo de replicación y reconstrucción de mensajes.

Tabla 5.2 Registros de tiempo reportados por el emisor y el receptor

Emisor	
`t_1`	Previo al envío de un mensaje desde el emisor.
`t_2`	Al recibir el mensaje de eco desde el receptor.
`t_3`	Después de la reconstrucción local del anclaje compartido.
Receptor	
`t_4`	Al recibir el mensaje del trazo transmitido por el emisor.
`t_5`	Después de completar el renderizado visual y la reconstrucción del anclaje del trazo recibido.

Dentro de ambas arquitecturas, cada trazo transmitido preserva su identificador de mensaje único en toda la retransmisión. Durante el postprocesamiento, estos identificadores se utilizaron para emparejar los eventos correspondientes en los registros del emisor y del receptor, permitiendo una alineación espacial coherente.

Este diseño aseguró que todas las métricas de latencia se derivaran de dominios de reloj internamente consistentes, eliminando imprecisiones debidas a desviaciones o desfases espaciales y temporales. En la Tabla 5.2 se presentan los registros de tiempo obtenidos por los dispositivos involucrados por cada trazo transmitido:

Partiendo de estas mediciones, se calcularon las siguientes métricas por cada prueba realizada:

- **Número de puntos espaciales:** P_n, requeridos para replicar un trazo.
- **Tiempo de respuesta desde el emisor:** $\Delta t_s = t_2 - t_1$, para evaluar el ciclo de transmisión.
- **Latencia de replicación:** $\Delta t_r = t_5 - t_4$, que aísla el retardo desde la recepción hasta la replicación visual.
- **Velocidad de recreación:** $\Delta t_r / P_n$, en puntos por segundo.

5.2. Resultados experimentales

El conjunto de resultados se analizó mediante estadística descriptiva (media, valor mínimo, valor máximo y desviación estándar). El análisis se centra en caracterizar el tiempo de respuesta del sistema y la sobrecarga de procesamiento, evaluando la capacidad de ambas arquitecturas de red (SRV–C y SRV–D) para soportar colaboración en tiempo real.

5.2.1. Latencia del sistema

Las Tablas 5.3 y 5.4 reportan los resultados en la transmisión Δt_s de ambas arquitecturas bajo seis tipos de pruebas realizadas, categorizadas por dos dispositivos y las figuras de prueba específicas. El Dispositivo A corresponde a un *smartphone* con Android 11 (Motorola One Hyper), equipado con un procesador Qualcomm Snapdragon 675, 4 GB de RAM y una pantalla FHD+ de 6.5 pulgadas, utilizado para las interacciones API–AR. El Dispositivo B, correspondiente a la configuración de un visor de realidad virtual Oculus Rift, fue conectado a un equipo de escritorio Alienware Aurora R7 con un CPU Intel Core i7-8700, 16 GB de RAM y una GPU NVIDIA GTX 1080, utilizado para las tareas API–VR.

En la arquitectura centralizada, los resultados revelan una disparidad fundamental de rendimiento entre las dos plataformas de hardware. El Dispositivo A exhibió no solo latencias promedio más altas, sino también una variabilidad significativamente mayor, como lo indican sus grandes desviaciones estándar (σ: 80.63 ms y 91.38 ms) y amplios rangos entre el valor mínimo y máximo de latencia registrado para todas las figuras. Por

ejemplo, para la prueba P_{Linea}, la latencia en el Dispositivo A varió entre 28 ms y 266 ms, mientras que para la prueba $P_{Espiral}$, la latencia varió entre 29 ms y 472 ms. Este patrón sugiere una volatilidad en el dispositivo móvil, probablemente debida a procesos del sistema operativo que se realizan en segundo plano, limitación térmica o asignación variable de recursos.

Tabla 5.3 Latencia de transmisión promedio de la arquitectura centralizada

Dispositivo	Prueba	$\bar{x}$ (ms)	min (ms)	max (ms)	σ (ms)	Peso (kB)
A	P_{Linea}	137.33	28	266	89.25	1.23
	$P_{Triangulo}$	139.90	25	352	87.30	2.89
	$P_{Cuadrado}$	160.96	31	354	81.53	3.19
	$P_{Pentagono}$	163.30	30	431	88.65	3.24
	$P_{Circulo}$	169.50	31	333	80.63	3.29
	$P_{Espiral}$	127.11	29	472	91.38	5.09
B	P_{Linea}	57.71	10	124	37.85	1.84
	$P_{Triangulo}$	73.20	11	189	40.95	4.61
	$P_{Cuadrado}$	63.51	10	178	40.44	3.95
	$P_{Pentagono}$	69.57	11	165	41.04	4.49
	$P_{Circulo}$	80.45	11	321	53.33	6.72
	$P_{Espiral}$	90.16	23	234	57.87	11.07

Así mismo, se observa una relación entre la geometría de las figuras y la latencia promedio en el Dispositivo A, donde algunas figuras geométricas como la `Linea` (137.33 ms) y el `Triángulo` (139.90 ms) presentan menores tiempos de respuesta comparado con figuras como el `Círculo` (169.50 ms) y el `Pentágono` (163.30 ms). Una excepción sucede con la `Espiral` (127.11 ms), que a pesar de tener el mayor tamaño de datos (5.09 Kb), registró la latencia promedio menor.

Por el contrario, el Dispositivo B demostró latencias consistentemente más bajas y estables. Sus desviaciones estándar fueron sustancialmente menores (entre 37.85 ms y 57.87 ms), y los rangos mínimo–máximo fueron más estrechos en todas las figuras, lo que indica una mejor respuesta debido a la configuración de hardware más potente. Este rendimiento puede visualizarse en la Figura 5.3, donde se contrasta la dispersión y medianas de ambos dispositivos.

En cambio, en la arquitectura descentralizada, se presenta una respuesta diferente entre ambos dispositivos. Para el Dispositivo A, se observa una mejora significativa en el rendimiento, con reducciones de latencia que superan el 75 % en comparación con la arquitectura centralizada. Las latencias promedio se mantienen en un rango notablemente

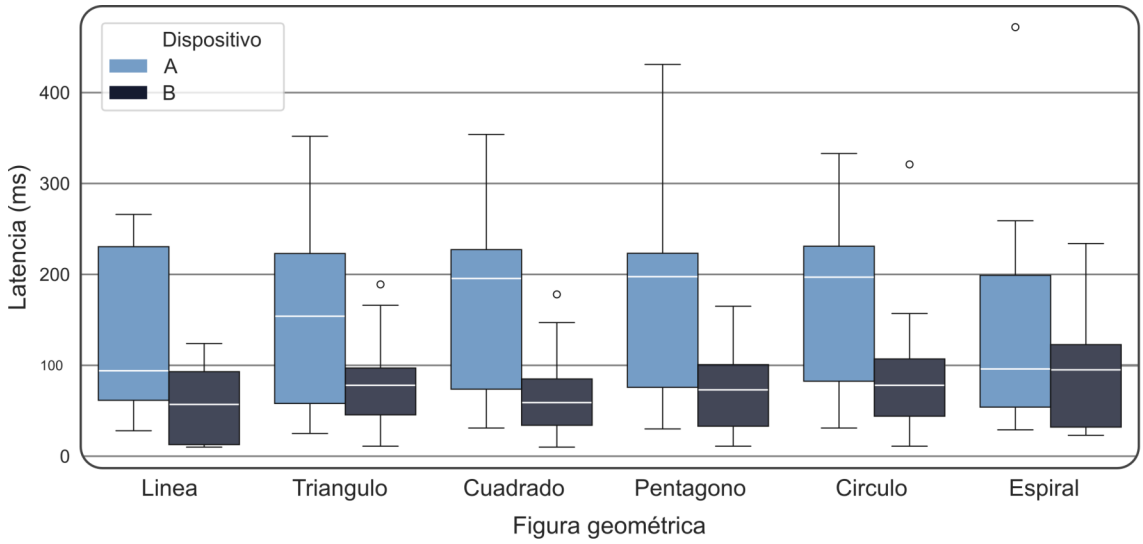


Figura 5.3: Boxplot de la latencia total del sistema para cada uno de los seis tipos de pruebas realizadas. El Dispositivo A corresponde a un smartphone Motorola One Hyper (cliente API–AR), mientras que el Dispositivo B designa un visor Oculus Rift conectado a un Alienware Aurora R7 (cliente API–VR).

estrecho (36.54 ms - 40.60 ms), con una variabilidad reducida significativamente (σ: 8.52-14.82 ms). Esta consistencia sugiere que la arquitectura descentralizada mitiga las fuentes de variabilidad presentes en la plataforma móvil.

La relación entre complejidad de figuras y latencia prácticamente desaparece en el Dispositivo A bajo la arquitectura descentralizada, indicando que el procesamiento distribuido minimiza los problemas de rendimiento asociados con figuras más complejas. Así mismo, el tamaño de los datos transmitidos muestra una reducción general en comparación con la arquitectura centralizada, mejorando adicionalmente el rendimiento de la red.

No obstante, para el Dispositivo B, los resultados presentan un comportamiento contraintuitivo. Las latencias promedio (76.20 ms - 97.55 ms) son consistentemente superiores a las observadas en la arquitectura centralizada para el mismo dispositivo. Este fenómeno sugiere una saturación por el proceso de coordinación en la arquitectura descentralizada, aunque la variabilidad se mantiene en rangos similares a la arquitectura centralizada.

Tabla 5.4 Latencia de transmisión promedio de la arquitectura descentralizada

Dispositivo	Prueba	$\bar{x}$ (ms)	min (ms)	max (ms)	σ (ms)	Peso (kB)
A	P_{Linea}	36.54	29	64	8.52	1.07
	$P_{Triangulo}$	38.56	24	70	11.64	2.32
	$P_{Cuadrado}$	39.75	28	99	14.82	2.47
	$P_{Pentagono}$	40.60	27	69	12.64	2.30
	$P_{Circulo}$	40.32	28	68	11.41	2.35
	$P_{Espiral}$	38.47	26	72	11.92	3.76
B	P_{Linea}	89.18	12	146	35.08	1.04
	$P_{Triangulo}$	87.21	22	145	31.04	2.60
	$P_{Cuadrado}$	97.55	22	153	30.15	2.92
	$P_{Pentagono}$	87.49	12	138	30.64	2.11
	$P_{Circulo}$	76.20	19	123	28.42	1.98
	$P_{Espiral}$	82.02	30	124	24.68	3.14

5.2.2. Latencia de replicación

Las Tablas 5.5 y 5.6 presentan la latencia de replicación Δt_r, que captura el retardo en el lado del usuario desde la recepción del mensaje hasta la finalización del renderizado visual. Esta métrica refleja la sobrecarga de procesamiento local una vez se completa la transmisión de datos, utilizando los mismos dos dispositivos descritos en la Sección 5.2.1.

En la arquitectura centralizada, el Dispositivo B demostró un rendimiento inferior en la latencia de replicación. Presentó latencias de replicación más altas en todas las figuras de prueba con promedios entre 9.77 ms y 11.23 ms, comparados con los valores registrados por el Dispositivo A, cuyos promedios oscilaban entre 1.94 ms y 3.50 ms. Sin embargo, el Dispositivo B mostró una estabilidad más notable, como lo evidencian las bajas desviaciones estándar registradas (σ: 2.18-3.64 ms) y rangos mínimo–máximo más estrechos que el Dispositivo A, lo que sugiere que, si bien el dispositivo móvil puede renderizar trazos muy rápido, es susceptible a retardos de procesamiento impredecibles, probablemente debido a la contención de recursos en el sistema Android, contrario al Dispositivo B que muestra un rendimiento más equilibrado.

En cambio, la implementación descentralizada demuestra mejoras significativas, particularmente en el Dispositivo B, donde se observa una reducción en las latencias de replicación. Los valores descienden a rangos extraordinariamente bajos, con múltiples figuras

(`Triángulo`, `Cuadrado`, `Pentágono`, `Espiral`) registrando latencias de replicación promedio de 1 ms y desviaciones estándar prácticamente nulas.

Tabla 5.5 Latencia de replicación promedio de la arquitectura centralizada

Dispositivo	Prueba	$\bar{x}$ (ms)	min (ms)	max (ms)	σ (ms)	N° de puntos
A	P_{Linea}	1.94	0	76	9.85	14.01
	$P_{Triangulo}$	3.50	0	49	10.29	36.64
	$P_{Cuadrado}$	2.21	0	76	10.18	41.60
	$P_{Pentagono}$	1.09	0	15	2.50	41.51
	$P_{Circulo}$	3.07	0	62	11.56	42.49
	$P_{Espiral}$	1.68	0	35	5.10	67.40
B	P_{Linea}	11.23	5	21	3.01	22.33
	$P_{Triangulo}$	10.94	5	16	2.58	60.97
	$P_{Cuadrado}$	10.89	3	19	2.90	51.82
	$P_{Pentagono}$	10.73	6	18	2.18	59.38
	$P_{Circulo}$	9.77	2	21	3.64	90.55
	$P_{Espiral}$	10.44	3	19	3.07	152.05

Tabla 5.6 Latencia de replicación promedio de la arquitectura descentralizada

Dispositivo	Prueba	$\bar{x}$ (ms)	min (ms)	max (ms)	σ (ms)	N° de puntos
A	P_{Linea}	2.23	0	48	7.19	11.73
	$P_{Triangulo}$	3.87	0	85	14.25	28.87
	$P_{Cuadrado}$	1.98	0	55	7.14	31.18
	$P_{Pentagono}$	1.01	0	24	3.07	28.70
	$P_{Circulo}$	0.60	0	9	1.24	29.36
	$P_{Espiral}$	2.72	0	82	12.00	48.57
B	P_{Linea}	0.016	0	1	0.13	11.18
	$P_{Triangulo}$	0.00	0	0	0.00	32.32
	$P_{Cuadrado}$	0.00	0	0	0.00	36.63
	$P_{Pentagono}$	0.00	0	0	0.00	25.74
	$P_{Circulo}$	0.037	0	2	0.27	23.79
	$P_{Espiral}$	0.00	0	0	0.00	39.67

Para el Dispositivo A, la arquitectura descentralizada mantiene latencias promedio similares a la configuración centralizada (0.60–3.87 ms), pero con una reducción general en su variación. Figuras como el `Círculo` muestran una mejora particularmente notable, con una latencia promedio de 0.60 ms y desviación estándar de solo 1.24 ms, comparado con 3.07 ms y 11.56 ms respectivamente en la arquitectura centralizada.

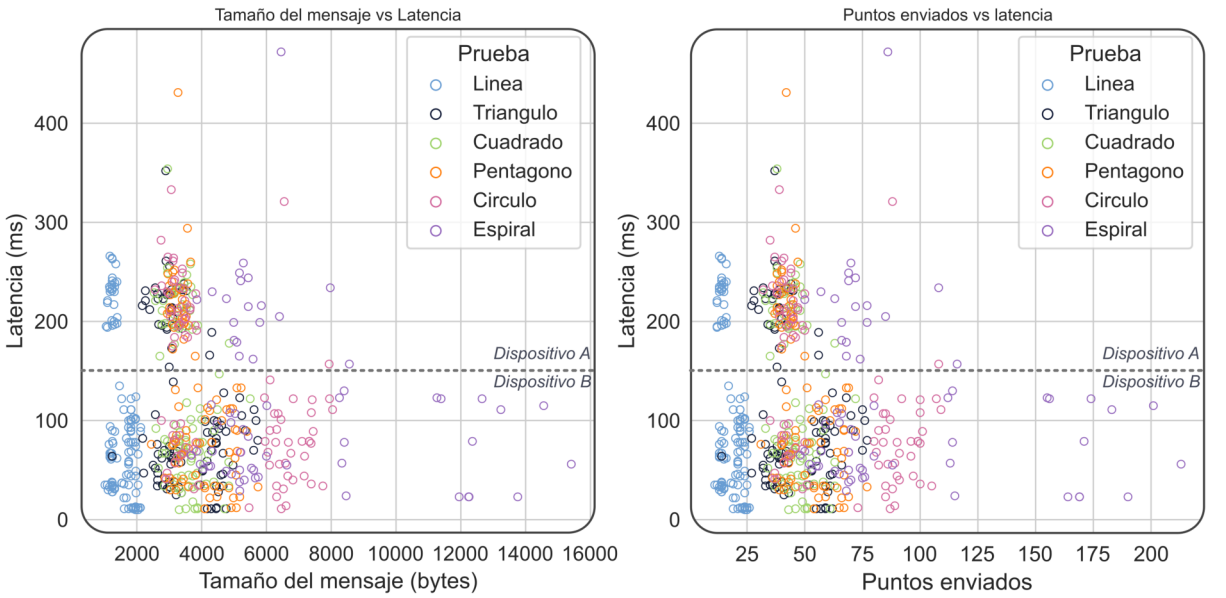
Cabe destacar que la relación entre la complejidad de la figura (tamaño del mensaje y número de puntos) y la latencia de replicación es débil para ambos dispositivos, como se muestra en la Figura 5.4. El Dispositivo B mantiene latencias estables a pesar del aumento en el número de puntos, tanto en la arquitectura centralizada como descentralizada, mientras que el Dispositivo A no muestra un comportamiento homogéneo entre la complejidad y la latencia promedio. Sin embargo, todas las latencias de replicación medidas se mantuvieron muy por debajo de los 15 ms en promedio, valor imperceptible para una operación de retroalimentación visual.

De acuerdo a los resultados de ambas arquitecturas, la descentralización revela una dependencia de la plataforma objetivo. Para las pruebas con dispositivos móviles (Dispositivo A), la arquitectura descentralizada provee mejores resultados en reducción de latencia. En cambio, para la configuración del Dispositivo B, la arquitectura centralizada asegura un mejor desempeño relacionado por el poder computacional del equipo y su optimización para los estándares de comunicación utilizados.

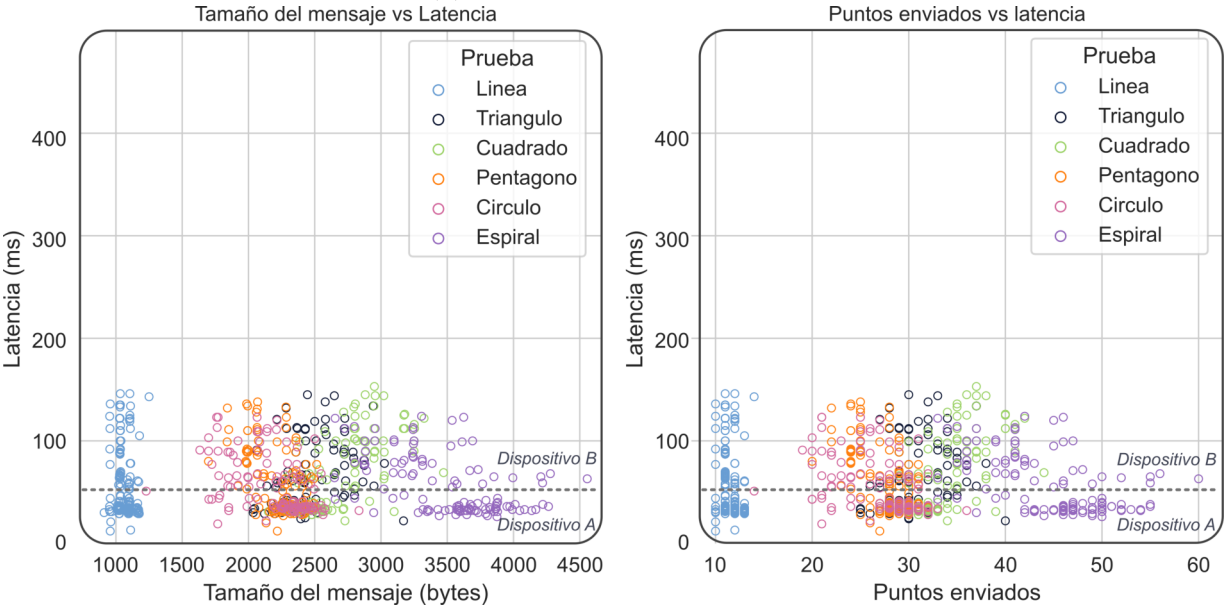
Respecto al análisis entre el tamaño de datos y su relación con la latencia se demuestra que la arquitectura descentralizada separa efectivamente estos parámetros en el dispositivo móvil, mientras que en la arquitectura centralizada aún se mantienen relacionados.

La comparación también revela que la descentralización proporciona mejores beneficios asimétricos. Para el Dispositivo B, la mejora es cuantitativamente superior, con reducciones de hasta el 70 % en latencia para la gran mayoría de las pruebas, resultado que permite presentar a la arquitectura descentralizada como una solución más adecuada para la reducción de los tiempos de procesamiento y la transmisión de mensajes, a pesar de que para el Dispositivo A las mejoras fueron más sutiles.

Finalmente, los tiempos de replicación medidos se mantuvieron muy por debajo del umbral de la percepción humana, confirmando la capacidad de ambas arquitecturas para proporcionar una retroalimentación visual instantánea. No obstante, la superior consistencia de la arquitectura descentralizada, principalmente en los resultados del Dispositivo B, sugiere una experiencia de usuario más fluida y predecible, libre de los microretardos ocasionales observados en la configuración centralizada.



(a) Arquitectura centralizada



(b) Arquitectura descentralizada

Figura 5.4: Comparación de latencias entre las distintas figuras geométricas bajo diferentes tamaños de mensaje y densidades de puntos. En (a) se visualizan los resultados obtenidos bajo la arquitectura centralizada, mientras que (b) presenta los correspondientes a la arquitectura descentralizada. En ambos casos, se añadió una línea divisoria aproximada que clasifica los resultados obtenidos de los dos dispositivos.

Capítulo 6

Conclusiones generales y perspectivas

Este capítulo presenta las conclusiones generales de la investigación. Se aborda la contribución que se realiza a la literatura y se finaliza con la discusión sobre investigaciones y trabajos futuros.

En esta tesis se presentó una comparativa entre el sistema SRV–C, una arquitectura centralizada y SRV–D, una arquitectura descentralizada para colaboración en tiempo real y colocalizada entre experiencias virtuales y aumentadas. Ambos sistemas fueron diseñados para operar completamente sobre redes locales sin depender de infraestructura propietaria. El sistema integró un mecanismo de sincronización, un protocolo de comunicación independiente de la serialización y APIs específicas para RA móvil y RV a escala habitación. La evaluación experimental de latencia de transmisión y replicación confirmó que la colaboración, haciendo uso de arquitecturas centralizadas, puede ser suficiente para experiencias simples que requieren una consistencia espacial entre dispositivos heterogéneos, sobre todo en aquellas actividades que requieren cargas de trabajo mínimas o moderadas.

Un hallazgo interesante en este análisis fue que los recursos de hardware si impactan significativamente el comportamiento del sistema. Los resultados revelaron una clara asimetría: la configuración de RV (Dispositivo B), quién presentó una capacidad de procesamiento superior, ofreció una latencia del sistema consistentemente menor y más estable, totalmente opuesta a los resultados obtenidos desde el dispositivo de RA móvil (Dispositivo A), quien exhibió mayor volatilidad a pesar de lograr menores tiempos promedio de replicación. Esto abre nuevas oportunidades de estudio donde se pueda experimentar con equipos de iguales capacidades de procesamiento, en lugar de intentar igualar o equiparar las capacidades heterogéneas de los dispositivos, siendo interesante cuestionarse el grado de relevancia que presenta la heterogeneidad en este tipo de experiencias.

Así mismo, los resultados globales refuerzan la capacidad de las configuraciones descentralizadas para satisfacer las necesidades de colaboración en XR. Además, validan el principio de diseño de adoptar la asimetría interactiva, permitiendo una participación diferenciada por dispositivo según sus capacidades técnicas, guiando nuestra hipótesis de que la descentralización es más que viable para la generación de nuevas experiencias XR compartidas.

Respecto a nuestros objetivos planteados inicialmente, se consiguió el diseño y evaluación satisfactoria de un entorno distribuido con soporte multimodal y diversidad de dispositivos, lo cual plantea nuevas líneas de exploración técnica. En trabajos futuros, se pretende atender las restricciones operativas identificadas en esta investigación. Como primer paso, se implementarán estrategias de sincronización dinámica, por ejemplo, colas de eventos priorizadas en función del tópico en donde se publique, modular la frecuencia de emisión de mensajes para evitar una saturación de la red y dar soporte a grupos de usuarios más grandes a través de implementaciones Web y de blockchain, con las que se permita diversificar geográficamente la experiencia. Así mismo, se realizarán estudios con distintos tipos de interacción y se explorarán soluciones de transmisión de mensajes parciales, sustituyendo el envío de paquetes íntegros por fragmentos parciales, sin incurrir en bloqueos o colisión entre los mismos, lo que implica la incorporación de mecanismos de consenso como Gossip, RAFT o Paxos, que no fueron implementados debido a la intención de mantener una comparación equiparable entre ambas arquitecturas. Finalmente, establecer un esquema compacto de serialización más sofisticado podría abrir oportunidades a establecer nuevos estándares de transmisión, por lo que se buscará realizar comparativas más robustas entre serializadores de entornos XR.

Estos esfuerzos estarán guiados por el objetivo de transformar el sistema SRV–D de un mero prototipo de investigación en una infraestructura extensible con potencial de despliegue en entornos reales.

Trabajos generados durante el desarrollo del proyecto

- G. A. Murillo Gutierrez, R. Jin, J.-P. I. Ramirez-Paredes, and U. H. Hernandez Belmonte, "A framework designed with perceptual symmetry and interactive asymmetry for XR collaboration," Symmetry, vol. 17, no. 11, 2025.
- G. A. Murillo Gutierrez, R. Jin, J.-P. I. Ramirez-Paredes, and U. H. Hernandez Belmonte, "ARCanvas: A Mobile-Based Collaborative Colocated AR Drawing Application," Communications in Computer and Information Science, vol. 2553, 2025.
- G. A. Murillo Gutierrez, R. Jin, J.-P. I. Ramirez-Paredes, and U. H. Hernandez Belmonte, "A Framework for Collaborative Augmented Reality Applications," In Companion Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, article 2, 1–2, 2025.
- U. H. Hernandez Belmonte, J.-P. I. Ramirez-Paredes, and G. A. Murillo Gutierrez, "Diseño e Implementación de una Plataforma de Realidad Aumentada Colaborativa," Avances en sistemas Mecatrónicos, vol. 1, no.1, 2023.

Bibliografía

[1] P. Wang, X. Bai, M. Billinghurst, S. Zhang, X. Zhang, S. Wang, W. He, Y. Yan, and H. Ji, "AR/MR remote collaboration on physical tasks: A review," *Robotics and Computer-Integrated Manufacturing*, vol. 72, dec 2021.

[2] J. Jang, Y. Ko, W. S. Shin, and I. Han, "Augmented reality and virtual reality for learning: An examination using an extended technology acceptance model," *IEEE Access*, vol. 9, pp. 6798–6809, 2021.

[3] A. Vidal Balea, O. Blanco Novoa, P. Fraga Lamas, M. Vilar Montesinos, and T. M. Fernández Caramés, "Creating collaborative augmented reality experiences for industry 4.0 training and assistance applications: Performance evaluation in the shipyard of the future," *Applied Sciences*, vol. 10, no. 24, 2020.

[4] L. Muñoz Saavedra, L. Miró Amarante, and M. Domínguez Morales, "Augmented and virtual reality evolution and future tendency," *Applied Sciences*, vol. 10, no. 1, 2020.

[5] P. Milgram, H. Takemura, A. Utsumi, and F. Kishino, "Augmented reality: A class of displays on the reality-virtuality continuum," in *Telemanipulator and Telepresence Technologies* (H. Das, ed.), vol. 2351, pp. 282–292, International Society for Optics and Photonics, SPIE, 1995.

[6] G. Lampropoulos, P. Fernández-Arias, A. de Bosque, and D. Vergara, "Virtual reality in engineering education: A scoping review," *Education Sciences*, vol. 15, no. 8, 2025.

[7] S. Y. Andalib, M. Monsur, C. Cook, M. Lemon, P. Zawarus, and L. Loon, "Enhancing landscape architecture construction learning with extended reality (XR): Comparing interactive virtual reality (VR) with traditional learning methods," *Education Sciences*, vol. 15, no. 8, 2025.

[8] J. Halman, S. Tencer, and M. Siemiński, "Artificial intelligence and extended reality in the training of vascular surgeons: A narrative review," *Medical Sciences*, vol. 13, no. 3, 2025.

[9] C. Hwang, T. Feuchtner, I. Oakley, and K. Grønbæk, "Enriching industrial training experience in virtual reality with pseudo-haptics and vibrotactile stimulation," in

Proceedings of the 30th ACM Symposium on Virtual Reality Software and Technology, VRST '24, (New York, NY, USA), Association for Computing Machinery, 2024.

[10] S. Zhang, Y. Li, K. L. Man, Y. Yue, and J. Smith, "Towards cross-reality interaction and collaboration: A comparative study of object selection and manipulation in reality and virtuality," in *2023 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pp. 330–337, 2023.

[11] T. Papadopoulos, K. Evangelidis, T. H. Kaskalis, G. Evangelidis, and S. Sylaiou, "Interactions in augmented and mixed reality: An overview," *Applied Sciences*, vol. 11, no. 18, 2021.

[12] A. Çöltekin, I. Lochhead, M. Madden, S. Christophe, A. Devaux, C. Pettit, O. Lock, S. Shukla, L. Herman, Z. Stachoň, P. Kubíček, D. Snopková, S. Bernardes, and N. Hedley, "Extended reality in spatial sciences: A review of research challenges and future directions," *ISPRS International Journal of Geo-Information*, vol. 9, no. 7, 2020.

[13] N. Numan and A. Steed, "Exploring user behaviour in asymmetric collaborative mixed reality," in *Proceedings of the 28th ACM Symposium on Virtual Reality Software and Technology*, VRST '22, (New York, NY, USA), Association for Computing Machinery, 2022.

[14] Y. Lee and B. Yoo, "XR collaboration beyond virtual reality: work in the real world," *Journal of Computational Design and Engineering*, vol. 8, pp. 756–772, 03 2021.

[15] J. Tümler, A. Toprak, and B. Yan, "Multi-user multi-platform xR collaboration: System and evaluation," in *Virtual, Augmented and Mixed Reality: Design and Development* (J. Y. C. Chen and G. Fragomeni, eds.), (Cham), pp. 74–93, Springer International Publishing, 2022.

[16] C. Lin, X. Sun, C. Yue, C. Yang, W. Gai, P. Qin, J. Liu, and X. Meng, "A novel workbench for collaboratively constructing 3D virtual environment," *Procedia Computer Science*, vol. 129, pp. 270–276, 2018. 2017 International Conference on Identification, Information and Knowledge in the Internet of Things.

[17] E. Games, "Photon engine," 2025. Consultado en julio de 2025.

[18] U. Technologies, "Netcode for gameobjects overview," 2025. Consultado en julio de 2025.

[19] M. Networking, "Mirror - networking library for unity," 2025. Consultado en julio de 2025.

[20] M. Corporation, "Playfab backend platform," 2025. Consultado en julio de 2025.

[21] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon, and J. C. Hart, "The cave: audio visual experience automatic virtual environment," *Commun. ACM*, vol. 35, p. 64–72, June 1992.

[22] C. Carlsson and O. Hagsand, "Dive a multi-user virtual reality system," in *Proceedings of IEEE Virtual Reality Annual International Symposium*, pp. 394–400, 1993.

[23] M. R. Macedonia, M. J. Zyda, D. R. Pratt, P. T. Barham, and S. Zeswitz, "Npsnet: A network software architecture for largescale virtual environments," *Presence: Teleoper. Virtual Environ.*, vol. 3, p. 265–287, Jan. 1994.

[24] IEEE Computer Society, "IEEE standard for distributed interactive simulation (dis) – communication services and profiles." IEEE Std 1278.2-2015, Nov. 2015. Disponible en: https://standards.ieee.org/ieee/1278.2/6202/.

[25] C. Greenhalgh and S. Benford, "MASSIVE: a distributed virtual reality system incorporating spatial trading," in *Proceedings of the 15th International Conference on Distributed Computing Systems (DCS'95)*, (Vancouver, Canada), pp. 27–34, IEEE Computer Society, 1995.

[26] T. Ohshima, K. Satoh, H. Yamamoto, and H. Tamura, "AR2 Hockey: A case study of collaborative augmented reality," in *Proceedings of the Virtual Reality Annual International Symposium*, VRAIS '98, (USA), p. 268, IEEE Computer Society, 1998.

[27] M. Billinghurst and H. Kato, "Collaborative mixed reality," in *Proceedings of the First International Symposium on Mixed Reality (ISMR)* (Y. Ohta and H. Tamura, eds.), (Yokohama, Japan), pp. 261–284, Springer-Verlag, 1999. ISBN 978-3-540-67260-6.

[28] M. Bauer, B. Bruegge, G. Klinker, A. MacWilliams, T. Reicher, S. Riss, C. Sandor, and M. Wagner, "Design of a component-based augmented reality framework," in *Proceedings IEEE and ACM International Symposium on Augmented Reality*, pp. 45–54, 2001.

[29] D. Schmalstieg, A. Fuhrmann, G. Hesina, Z. Szalavári, L. M. Encarnação, M. Gervautz, and W. Purgathofer, "The studierstube augmented reality project," *Presence*, vol. 11, no. 1, pp. 33–54, 2002.

[30] Vuforia, "Vuforia developer portal," 2021. Consultado en agosto de 2025.

[31] Apple Inc., "Arkit framework." https://developer.apple.com/documentation/arkit, 2017. Consultado en agosto de 2025.

[32] Google Inc., "Arcore sdk for augmented reality." https://developers.google.com/ar, 2018. Consultado en agosto de 2025.

[33] I. B. K. Manuaba, "Mobile based augmented reality application prototype for remote collaboration scenario using ARCore cloud anchor," *Procedia Computer Science*, vol. 179, pp. 289–296, 2021.

[34] P. Boonbrahm, C. Kaewrat, and S. Boonbrahm, "Effective collaborative design of large virtual 3D model using multiple AR markers," *Procedia Manufacturing*, 2020.

[35] Microsoft Corporation, "Microsoft mesh." https://learn.microsoft.com/en-us/mesh/, 2020. Consultado en agosto de 2025.

[36] Spatial Systems Inc., "Spatial creator toolkit (Unity SDK)." https://support.spatial.io/hc/en-us/articles/11633003556372-Spatial-Creator-Toolkit-Unity-SDK, 2023. Consultado en agosto de 2025.

[37] Meta Platforms Inc., "Meta XR core SDK." https://assetstore.unity.com/packages/tools/integration/meta-xr-core-sdk-269169, 2023. Consultado en agosto de 2025.

[38] Apple Inc., "Realitykit framework documentation." https://developer.apple.com/documentation/realitykit, 2025. Consultado en septiembre de 2025.

[39] Microsoft Corporation, "Mixed reality toolkit (MRTK) for Unity." https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk3-overview/, 2023. Consultado en agosto de 2025.

[40] U. Technologies, "XR interaction toolkit manual." https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@3.0/manual/index.html, 2025. Consultado en septiembre de 2025.

[41] G. A. Murillo Gutierrez, *Diseño, implementación y evaluación de un sistema de realidad aumentada colaborativa*. Tesis, Universidad de Guanajuato, Guanajuato, México, 2023. Repositorio Institucional de la Universidad de Guanajuato.

[42] G. A. Murillo Gutierrez, R. Jin, J.-P. I. Ramirez-Paredes, and U. H. Hernandez Belmonte, "A framework designed with perceptual symmetry and interactive asymmetry for XR collaboration," *Symmetry*, vol. 17, no. 11, 2025.

[43] Google, "ARCore supported devices," 2025. Consultado en julio de 2025.

[44] N. Stephenson, *Snow Crash: A Novel*. New York: Bantam Books, 1992.

[45] M. P. Inc., "Introducing Meta: A social technology company," 2021. Accedido el 6 de octubre de 2025.

[46] P. A. Rauschnabel, R. Felix, C. Hinsch, H. Shahab, and F. Alt, "What is XR? towards a framework for augmented and virtual reality," *Computers in Human Behavior*, vol. 133, p. 107289, 2022.

[47] S. Aukstakalnis, *Practical Augmented Reality*. Crawfordsville - Indiana: Mark L. Taub, 2016.

[48] S. Hudson, S. Matson-Barkat, N. Pallamin, and G. Jegou, "With or without you? interaction and immersion in a virtual reality experience," *Journal of Business Research*, vol. 100, pp. 459–468, 2019.

[49] I. Rakkolainen, A. Farooq, J. Kangas, J. Hakulinen, J. Rantala, M. Turunen, and R. Raisamo, "Technologies for multimodal interaction in extended reality—a scoping review," *Multimodal Technologies and Interaction*, vol. 5, no. 12, 2021.

[50] H. Schraffenberger and E. van der Heide, "Multimodal augmented reality: the norm rather than the exception," in *Proceedings of the 2016 Workshop on Multimodal Virtual and Augmented Reality*, MVAR '16, (New York, NY, USA), Association for Computing Machinery, 2016.

[51] M. Baxter, A. Bleakley, J. Edwards, L. Clark, B. R. Cowan, and J. R. Williamson, ""You, move there!": Investigating the impact of feedback on voice control in virtual environments," in *Proceedings of the 3rd Conference on Conversational User Interfaces*, CUI '21, (New York, NY, USA), Association for Computing Machinery, 2021.

[52] A. Adkins, R. Canales, and S. Jörg, "Hands or controllers? how input devices and audio impact collaborative virtual reality," in *Proceedings of the 30th ACM Symposium on Virtual Reality Software and Technology*, VRST '24, (New York, NY, USA), Association for Computing Machinery, 2024.

[53] K. Pfeuffer, H. Gellersen, and M. Gonzalez-Franco, "Design principles and challenges for gaze + pinch interaction in XR," *IEEE Computer Graphics and Applications*, vol. 44, no. 3, pp. 74–81, 2024.

[54] J. Kim, J. Cha, H. Lee, and S. Kim, "Hand-free natural user interface for VR HMD with IR based facial gesture tracking sensor," in *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology*, VRST '17, (New York, NY, USA), Association for Computing Machinery, 2017.

[55] P.-S. Ku, T.-Y. Wu, and M. Y. Chen, "EyeExpression: exploring the use of eye expressions as hands-free input for virtual and augmented reality devices," in *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology*, VRST '17, (New York, NY, USA), Association for Computing Machinery, 2017.

[56] O. Bau and I. Poupyrev, "REVEL: tactile feedback technology for augmented reality," *ACM Trans. Graph.*, vol. 31, July 2012.

[57] J. Wang and S. Gao, "Electronic skin for virtual sensation generation in immersive virtual and augmented reality," *IEEE Open Journal on Immersive Displays*, vol. 1, pp. 1–8, 2024.

[58] A. S. Tanenbaum and M. van Steen, *Distributed Systems: Principles and Paradigms*, ch. 1, pp. 2–9. CreateSpace Independent Publishing Platform, 2 ed., 2016.

[59] A. S. Tanenbaum and M. van Steen, *Distributed Systems: Principles and Paradigms*, ch. 2, pp. 36–54. CreateSpace Independent Publishing Platform, 2 ed., 2016.

[60] S. A. Rollán, "Arquitectura del software: Análisis de napster, gnutella y skype." https://www.studocu.com/es/document/universidad-pontificia-de-salamanca/arquitectura-del-software/arquitectura-del-software-analisis-de-napster-gnutella-y-skype-20242025/135426806, 2024. Análisis comparativo de arquitecturas P2P.

[61] T. A. Alghamdi, R. Khalid, and N. Javaid, "A survey of blockchain based systems: Scalability issues and solutions, applications and future challenges," *IEEE Access*, vol. 12, pp. 79626–79651, 2024.

[62] J. Glazer and S. Madhav, *Multiplayer Game Programming: Architecting Networked Games*, ch. 2, pp. 17–19. Boston, MA: Addison-Wesley Professional, 2016.

[63] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Concepts*, ch. 17, pp. 760–765. Wiley, 10 ed., 2020.

[64] J. Glazer and S. Madhav, *Multiplayer Game Programming: Architecting Networked Games*, ch. 2, pp. 39–43. Boston, MA: Addison-Wesley Professional, 2016.

[65] J. Glazer and S. Madhav, *Multiplayer Game Programming: Architecting Networked Games*, ch. 3, pp. 65–68. Boston, MA: Addison-Wesley Professional, 2016.

[66] J. Glazer and S. Madhav, *Multiplayer Game Programming: Architecting Networked Games*, ch. 4, pp. 101–110. Boston, MA: Addison-Wesley Professional, 2016.

[67] J. Glazer and S. Madhav, *Multiplayer Game Programming: Architecting Networked Games*, ch. 5, pp. 139–148. Boston, MA: Addison-Wesley Professional, 2016.

[68] B. A. Forouzan, *Data Communications and Networking*, ch. 10, pp. 258–281. McGraw-Hill Education, 5 ed., 2013.

[69] J. Nielsen, "Response times: The 3 important limits," 1993. Consultado en julio de 2025.

[70] J. Glazer and S. Madhav, *Multiplayer Game Programming: Architecting Networked Games*, ch. 7, pp. 139–209. Boston, MA: Addison-Wesley Professional, 2016.

[71] J. Glazer and S. Madhav, *Multiplayer Game Programming: Architecting Networked Games*, ch. 11, pp. 280–286. Boston, MA: Addison-Wesley Professional, 2016.

[72] Unity Technologies, "Unity engine." https://unity.com, 2025. Versión utilizada: 2022.3 LTS.

[73] Exit Games, "Photon unity networking (PUN)." https://www.photonengine.com/pun, 2025. Middleware para redes multijugador en Unity.

[74] Epic Games, "Unreal engine." https://www.unrealengine.com, 2025. Motor de desarrollo con replicación nativa y soporte multijugador.

[75] Godot Engine Contributors, "Godot engine." https://godotengine.org, 2025. Motor libre y abierto para desarrollo 2D/3D con capacidades de red.

[76] K. H. Ahlers, A. Kramer, D. E. Breen, P.-Y. Chevalier, C. Crampton, E. Rose, M. Tuceryan, R. T. Whitaker, and D. Greer, "Distributed augmented reality for collaborative design applications," *Computer Graphics Forum*, vol. 14, no. 3, pp. 3–14, 1995.

[77] G. M. Olson and J. S. Olson, "Distance matters," *Human–Computer Interaction*, vol. 15, no. 2-3, pp. 139–178, 2000.

[78] A. Schäfer, G. Reis, and D. Stricker, "A survey on synchronous augmented, virtual, andmixed reality remote collaboration systems," *ACM Comput. Surv.*, vol. 55, Dec. 2022.

[79] T. Braud, L.-H. Lee, A. Alhilal, C. B. Fernández, and P. Hui, "DiOS—an extended reality operating system for the metaverse," *IEEE MultiMedia*, vol. 30, no. 2, pp. 70–80, 2023.

[80] D. G. Brown, S. J. Julier, Y. Baillot, M. A. Livingston, and L. J. Rosenblum, "Event-based data distribution for mobile augmented reality and virtual environments," *Presence*, vol. 13, no. 2, pp. 211–221, 2004.

[81] J. Herskovitz, Y. F. Cheng, A. Guo, A. P. Sample, and M. Nebeling, "XSpace: An augmented reality toolkit for enabling spatially-aware distributed collaboration," *Proc. ACM Hum.-Comput. Interact.*, vol. 6, no. ISS, 2022.

[82] A. A. Simiscuka, T. M. Markande, and G.-M. Muntean, "Real-virtual world device synchronization in a cloud-enabled social virtual reality IoT network," *IEEE Access*, vol. 7, pp. 106588–106599, 2019.

[83] A. Guo, I. Canberk, H. Murphy, A. Monroy-Hernández, and R. Vaish, "Blocks: Collaborative and persistent augmented reality experiences," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 3, Sept. 2019.

[84] V. Pereira, T. Matos, R. Rodrigues, R. Nóbrega, and J. Jacob, "Extended reality framework for remote collaborative interactions in virtual environments," in *2019 International Conference on Graphics and Interaction (ICGI)*, pp. 17–24, 2019.

[85] G. Kostov and J. Wolfartsberger, "Designing a framework for collaborative mixed reality training," *Procedia Computer Science*, 2022.

[86] H.-J. Guo, O. E. Ashtiani, and B. Prabhakaran, "Experiences with CAMRE: Single-device collaborative adaptive mixed reality environment," *ArXiv*, vol. abs/2310.04996, 2023.

[87] B. Di Martino, G. J. Pezzullo, V. Bombace, L.-H. Li, and K.-C. Li, "On exploiting and implementing collaborative virtual and augmented reality in a cloud continuum scenario," *Future Internet*, vol. 16, no. 11, 2024.

[88] D. Mourtzis, V. Siatras, J. Angelopoulos, and N. Panopoulos, "An augmented reality collaborative product design cloud-based platform in the context of learning factory," *Procedia Manufacturing*, vol. 45, pp. 546–551, 1 2020.

[89] I. Viola, J. Jansen, S. Subramanyam, I. Reimat, and P. Cesar, "VR2Gather: A collaborative, social virtual reality system for adaptive, multiparty real-time communication," *IEEE MultiMedia*, vol. 30, no. 2, pp. 48–59, 2023.

[90] B. Han, P. Pathak, S. Chen, and L.-F. Yu, "CoMIC: A collaborative mobile immersive computing infrastructure for conducting multi-user XR research," *IEEE Network*, vol. 37, no. 6, pp. 124–131, 2023.

[91] M. Sereno, X. Wang, L. Besancon, M. J. Mcguffin, and T. Isenberg, "Collaborative work in augmented reality: A survey," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 10 2020.

[92] V. Bréhault, E. Dubois, A. Prouzeau, and M. Serrano, "A systematic literature review to characterize asymmetric interaction in collaborative systems," in *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, CHI '25, (New York, NY, USA), Association for Computing Machinery, 2025.

[93] J. G. Grandi, H. G. Debarba, and A. Maciel, "Characterizing asymmetric collaborative interactions in virtual and augmented realities," in *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 127–135, 2019.

[94] A. Agnès, F. Sylvain, R. Vanukuru, and S. Richir, "Studying the effect of symmetry in team structures on collaborative tasks in virtual reality," *Behaviour & Information Technology*, vol. 42, no. 14, pp. 2467–2475, 2023.

[95] Y. Huang, H. Wang, X. Qiao, X. Su, Y. Li, S. Dustdar, and P. Zhang, "SCAXR: Empowering scalable multi-user interaction for heterogeneous XR devices," *IEEE Network*, vol. 38, no. 4, pp. 250–258, 2024.

[96] D. Frey, J. Royan, R. Piegay, A.-M. Kermarrec, E. Anceaume, and F. Le Fessant, "Solipsis: A decentralized architecture for virtual environments," in *1st International Workshop on Massively Multiuser Virtual Environments*, (Reno, NV, United States), Mar. 2008.

[97] S. Huh, S. Muralidharan, H. Ko, and B. Yoo, "XR collaboration architecture based on decentralized web," in *Proceedings of the 24th International Conference on 3D Web Technology*, Web3D '19, (New York, NY, USA), p. 1–9, Association for Computing Machinery, 2019.

[98] N. Suslov, "Implementing decentralized virtual time in p2p collaborative learning environment for web XR," in *2021 7th International Conference of the Immersive Learning Research Network (iLRN)*, pp. 1–4, 2021.

[99] C. Corporation, "Croquet OS: A shared reality operating system." https://croquet.io, 2020. Consultado en octubre de 2025.

[100] M. Norman, G. Lee, R. T. Smith, and M. Billinghurs, "A mixed presence collaborative mixed reality system," in *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 1106–1107, 2019.

[101] N. Pereira, A. Rowe, M. W. Farb, I. Liang, E. Lu, and E. Riebling, "ARENA: The augmented reality edge networking architecture," in *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 479–488, 2021.

[102] Decentraland, "About decentraland." https://docs.decentraland.org/player/general/about/, 2025. Consultado el 14 de octubre de 2025.

[103] A. Ghosh, Lavanya, V. Hassija, V. Chamola, and A. El Saddik, "A survey on decentralized metaverse using blockchain and web 3.0 technologies, applications, and more," *IEEE Access*, vol. 12, pp. 146915–146948, 2024.

[104] S. K. Jagatheesaperumal, K. Ahmad, A. Al-Fuqaha, and J. Qadir, "Advancing education through extended reality and internet of everything enabled metaverses: Applications, challenges, and open issues," *IEEE Transactions on Learning Technologies*, vol. 17, pp. 1120–1139, 2024.

[105] H. Zhang, L. Li, Q. Lu, Y. Yue, Y. Huang, and S. Dustdar, "Distributed realtime rendering in decentralized network for mobile web augmented reality," *Future Generation Computer Systems*, vol. 158, pp. 530–544, 2024.

[106] S. Sharma, D. Das, and S. Chaudhury, "A decentralized privacy-preserving XR system for 3D medical data visualization using hybrid biometric cryptosystem," *Scientific Reports*, vol. 15, no. 28568, 2025.

[107] P. Bhattacharya, D. Saraswat, A. Dave, M. Acharya, S. Tanwar, G. Sharma, and I. E. Davidson, "Coalition of 6G and blockchain in AR/VR space: Challenges and future directions," *IEEE Access*, vol. 9, pp. 168455–168484, 2021.

[108] C. Baillard, M. Fradet, V. Alleaume, P. Jouet, and A. Laurent, "Multi-device mixed reality TV: a collaborative experience with joint use of a tablet and a headset," in *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology*, VRST '17, (New York, NY, USA), Association for Computing Machinery, 2017.

[109] T. Piumsomboon, A. Dey, B. Ens, G. Lee, and M. Billinghurst, "CoVAR: Mixed-platform remote collaborative augmented and virtual realities system with shared collaboration cues," in *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)*, pp. 218–219, 2017.

[110] W. Zhang, B. Han, P. Hui, V. Gopalakrishnan, E. Zavesky, and F. Qian, "CARS: Collaborative augmented reality for socialization," *HotMobile 2018 - Proceedings of the 19th International Workshop on Mobile Computing Systems and Applications*, vol. 2018-February, pp. 25–30, 2 2018.

[111] D. C. Rompapas, C. Sandor, A. Plopski, D. Saakes, J. Shin, T. Taketomi, and H. Kato, "Towards large scale high fidelity collaborative augmented reality," *Computers and Graphics (Pergamon)*, vol. 84, pp. 24–41, 11 2019.

[112] S. Benbelkacem, N. Zenati-Henda, D. Aouam, Y. Izountar, and S. Otmane, "MVC-3DC: Software architecture model for designing collaborative augmented reality and virtual reality systems," *Journal of King Saud University - Computer and Information Sciences*, vol. 32, pp. 433–446, 5 2020.

[113] A. Villanueva, Z. Zhu, Z. Liu, K. Peppler, T. Redick, and K. Ramani, "Meta-AR-App: An authoring platform for collaborative augmented reality in STEM classrooms," *Conference on Human Factors in Computing Systems - Proceedings*, 4 2020.

[114] T. Porcino, S. A. Ghaeinian, J. Franz, J. Malloch, and D. Reilly, "Design of an XR collab. arch. for mixed immersive and MS interaction," 2022.

[115] Y. Huang, H. Wang, X. Qiao, X. Su, Y. Li, S. Dustdar, and P. Zhang, "SCAXR: Empowering scalable multi-user interaction for heterogeneous XR devices," *IEEE Network*, vol. 38, no. 4, pp. 250–258, 2024.

[116] H. Neeli, K. Q. Tran, J. D. Velazco-Garcia, and N. V. Tsekos, "A multiuser, multisite, and platform-independent on-the-cloud framework for interactive immersion in holographic XR," *Applied Sciences*, vol. 14, no. 5, 2024.

[117] A. Close, S. Field, and R. Teather, "Visual thinking in virtual environments: evaluating multidisciplinary interaction through drawing ideation in real-time remote co-design," *Frontiers in Virtual Reality*, vol. 4, 2023.

[118] G. A. Murillo Gutierrez, "SRVS-C: Spatially referenced virtual synchronization for collaboration." https://github.com/MurilloLog/SRVS-C, 2025. GitHub repository. Framework for collaborative mixed reality experiences integrating AR and VR.

[119] A. Nagy and B. Kovari, "Analyzing .NET serialization components," in *2016 IEEE 11th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, pp. 425–430, 2016.

[120] N. Contributors, "NetMQ - native C# port of ZeroMQ." https://netmq.readthedocs.io/en/latest/, 2025. Consultado en septiembre de 2025.

[121] F. Reyes-Aviles, P. Fleck, D. Schmalstieg, and C. Arth, "Compact world anchors: Registration using parametric primitives as scene description," *IEEE Transactions on Visualization & Computer Graphics*, vol. 29, pp. 4140–4153, Oct. 2023.

[122] J. J. Dudley, H. Schuff, and P. O. Kristensson, "Bare-handed 3D drawing in augmented reality," in *Proceedings of the 2018 Designing Interactive Systems Conference*, DIS '18, (New York, NY, USA), p. 241–252, Association for Computing Machinery, 2018.