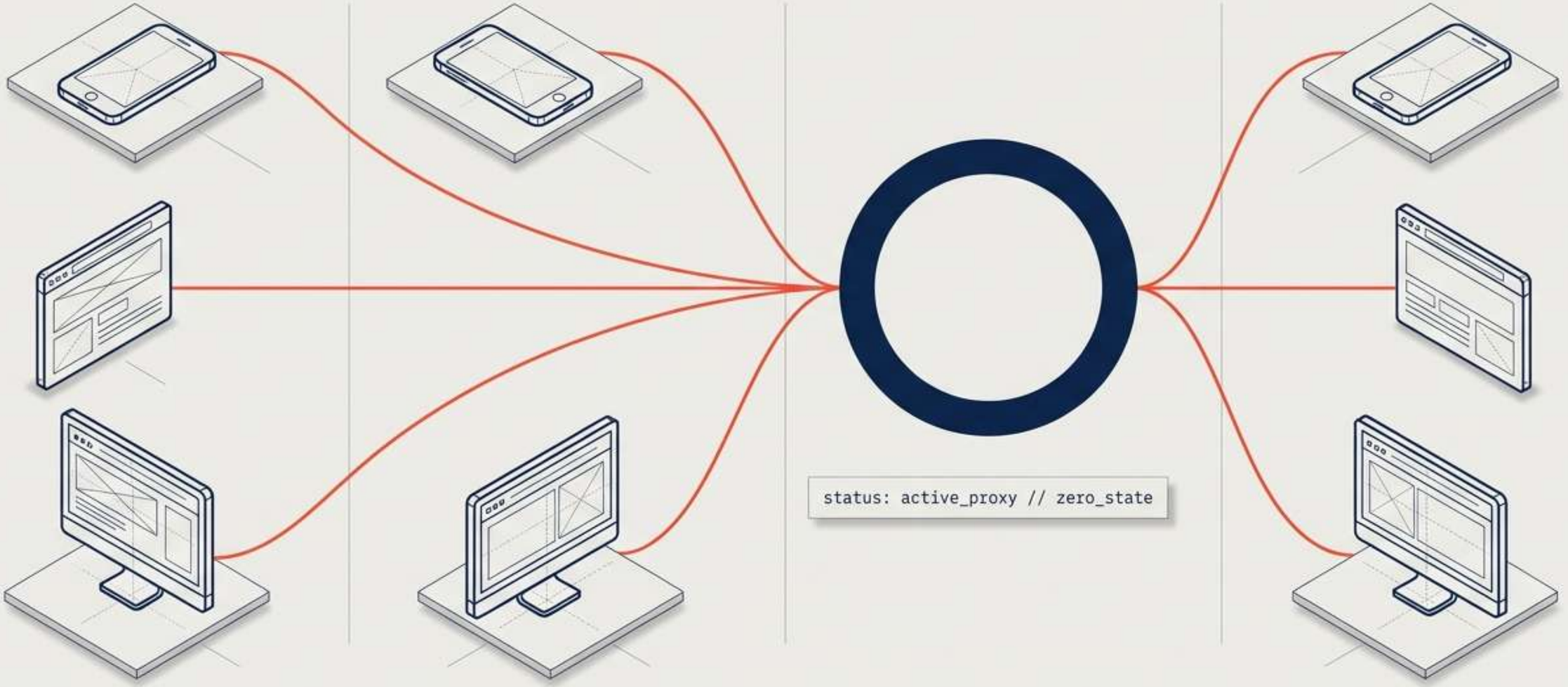
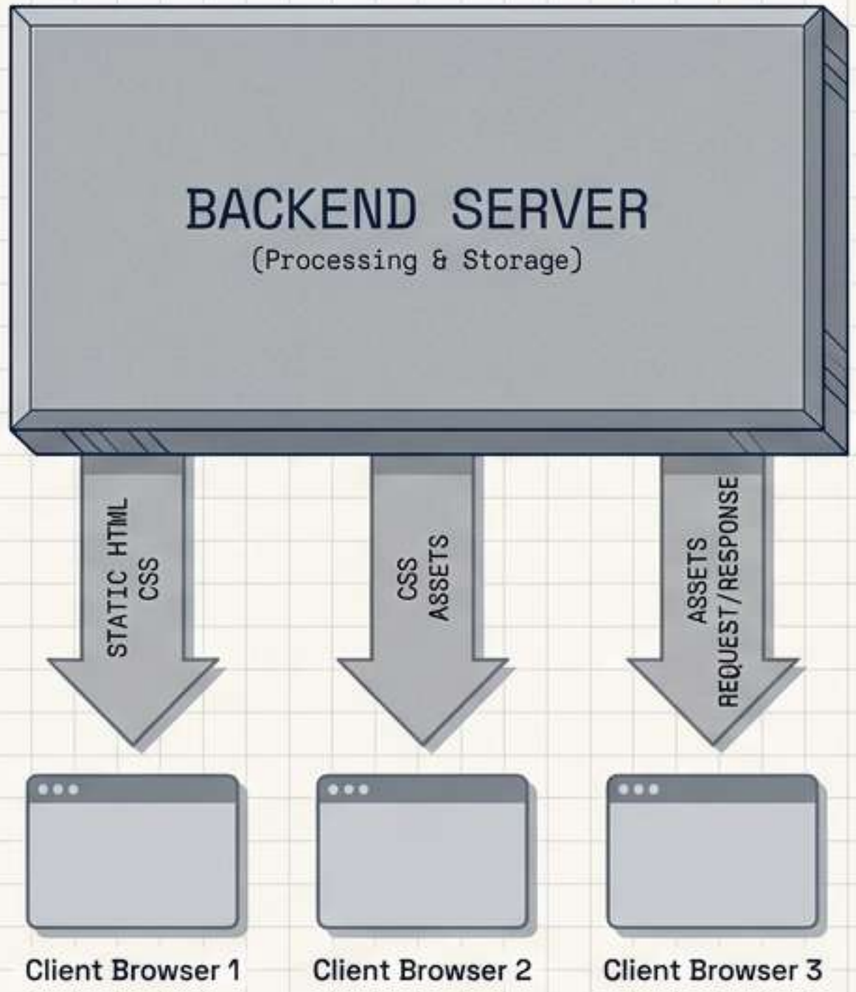
RCWeb Live: The Real-Time Distributed Web

Redefining the browser as both server and client.

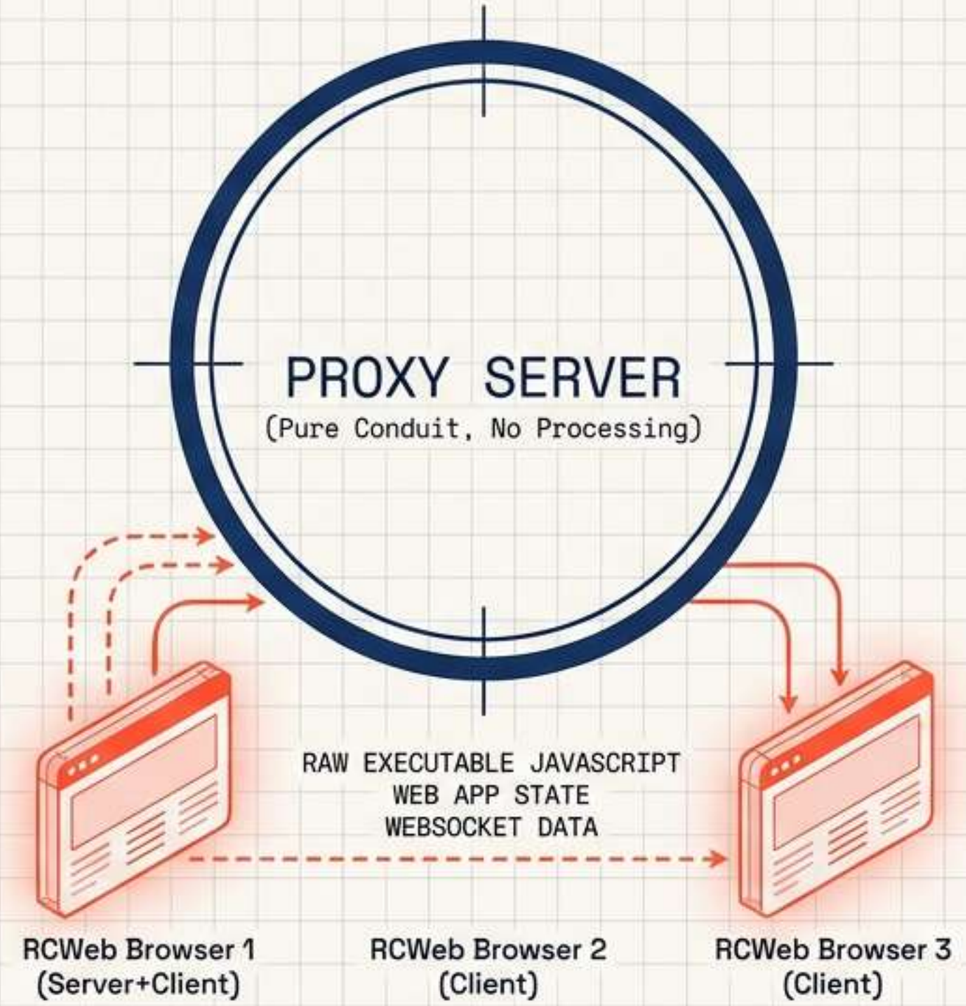


Bypassing the server to share JavaScript in real-time

TRADITIONAL MODEL



RCWEB MODEL



The Dual-Role Browser

Every web browser running an RCWeb app operates simultaneously as a web server and a web client.

Raw JS Sharing

Web apps update by transferring raw JavaScript between browsers over firewall-resistant persistent WebSocket connections.

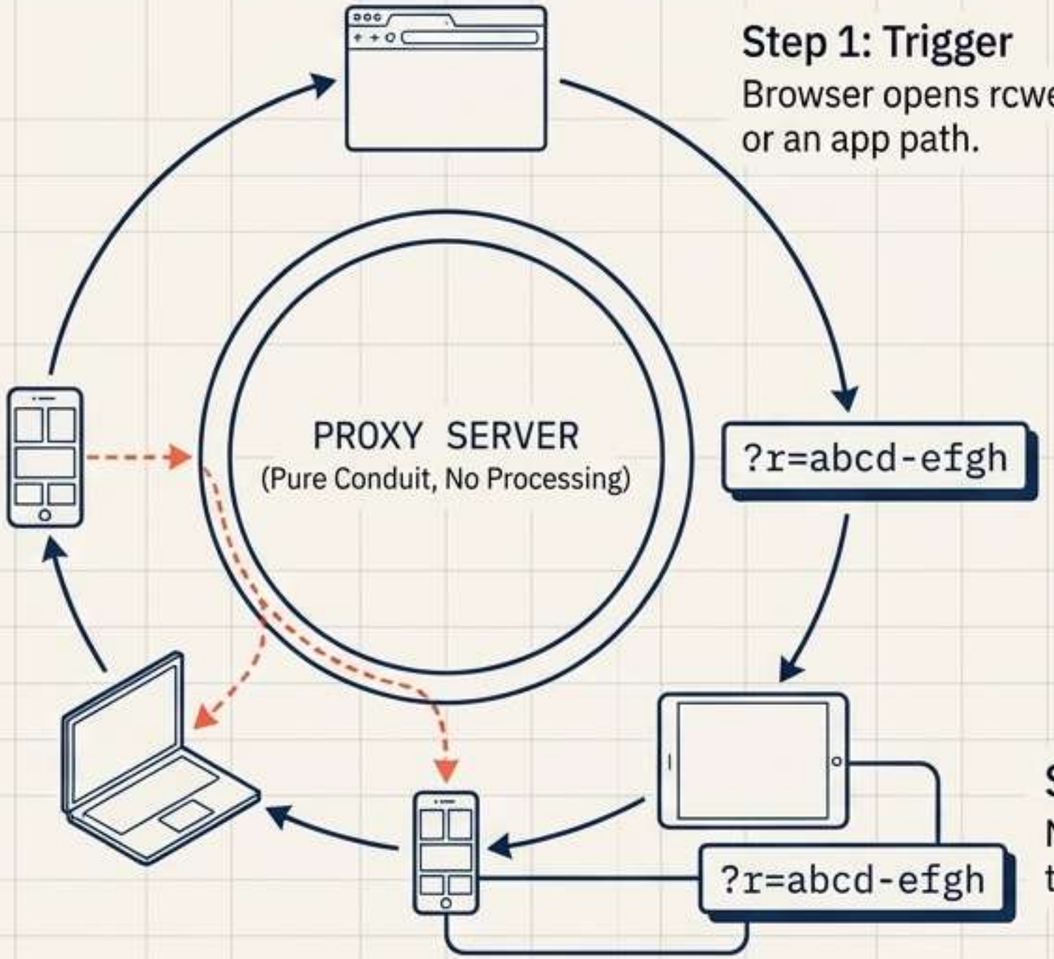
Zero Server Processing

The central server proxies data as raw bytes without parsing, processing, or storing any application data.

The architectural pivot to zero-overhead infrastructure

Dimension	Traditional Server-Side	RCWeb Client-Side
Resource Usage (CPU/Memory)	Extremely High	**Negligible Footprint**
System Responsiveness	Delayed by processing	**Instant execution**
Application State	Locked in Server	**Shared strictly Client-side**
Database & Storage	Mandatory, requires backups	**None** required; no data stored online
User Privacy	Compromised by central storage	**Absolute**; data disappears when session ends

The Virtual Room orchestrates the stateless mesh



Step 1: Trigger
Browser opens rcweb.live or an app path.

Step 2: Isolation
The server isolates the session by generating a unique room ID.

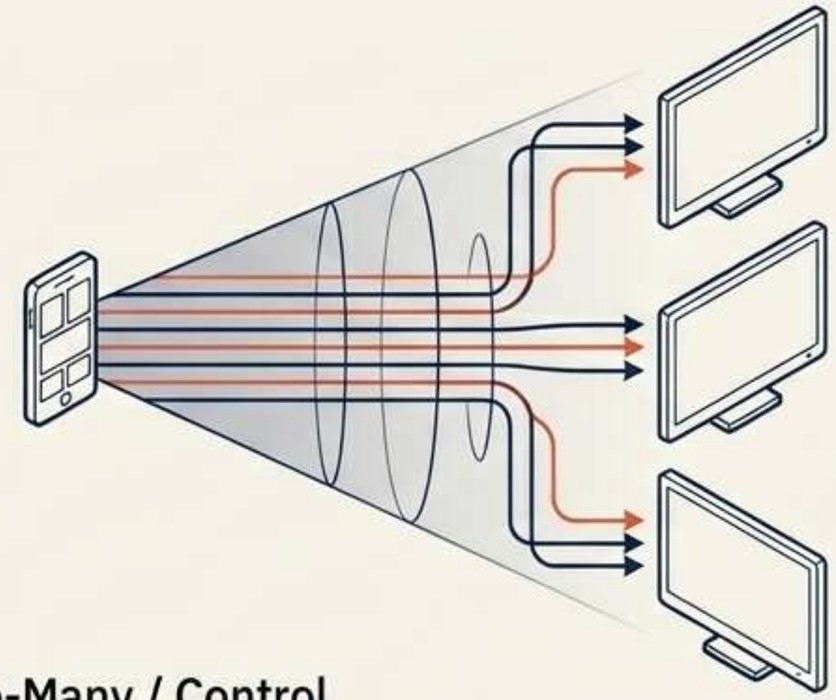
Step 3: Convergence
Multiple devices connect using the identical parameter.

Step 4: Execution
WebSocket messages containing executable JS are broadcast to all clients.

Apps provide the initial state; RCWeb facilitates the communication. No server-side code changes are ever required to launch a new app.

Two fundamental patterns drive distributed interactions

The Asymmetric Pattern

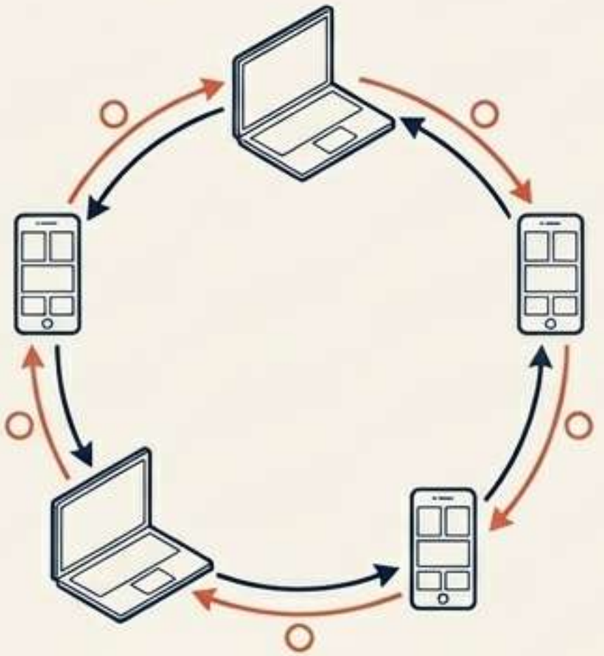


1-to-Many / Control

Remote Control & Broadcasting

Powers games, camera broadcasting, and digital signage.

The Symmetric Pattern

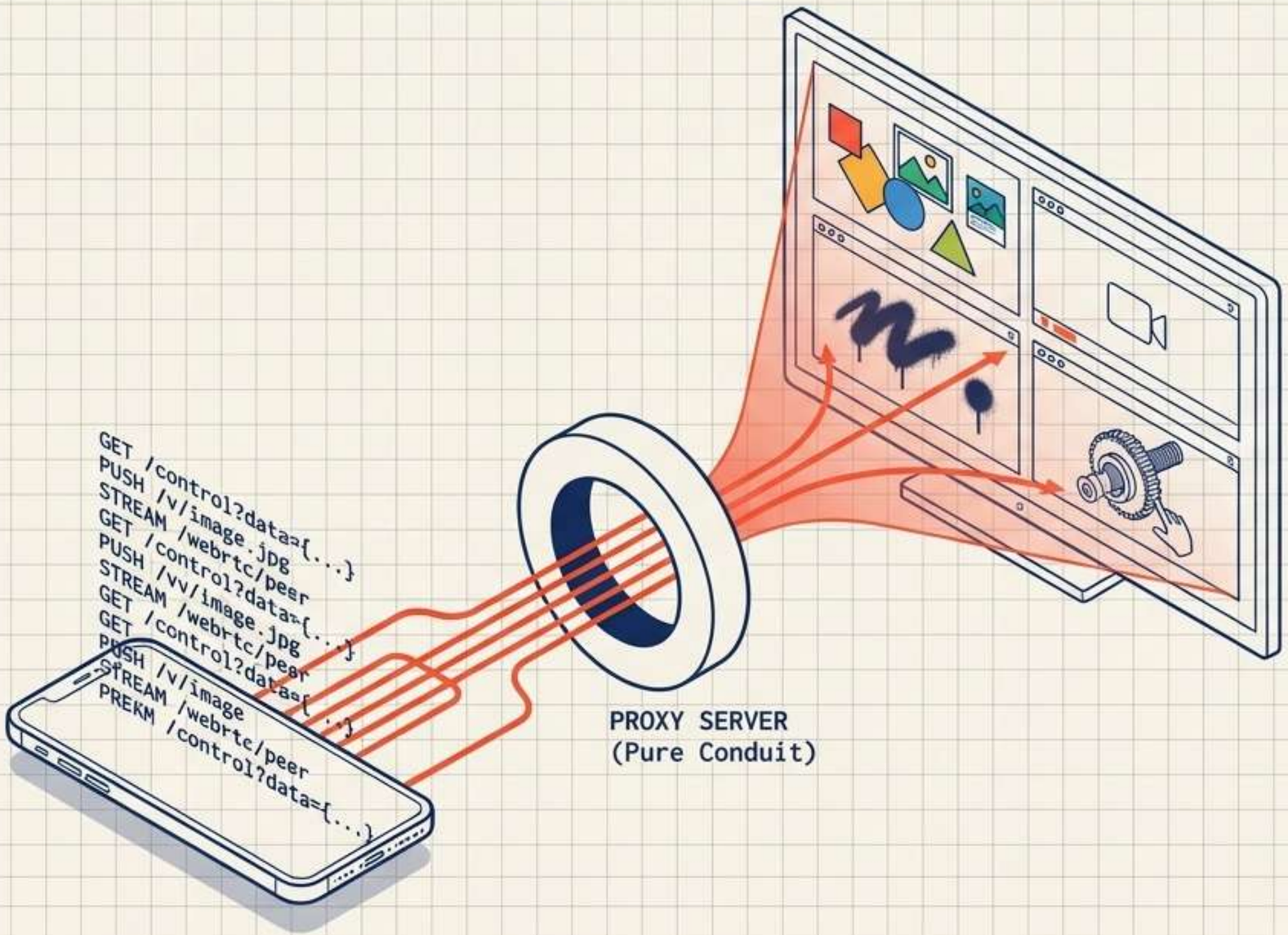


Many-to-Many / Collaborate

Decentralised Collaboration

Powers zero-database group chat, multi-gigabyte file sharing, and whiteboards.

Transforming smartphones into universal remote controllers



Viewer & Control (/v/, /c/)

Push colours, images, videos, and raw HTML/JS instantly from a mobile device to a large display.

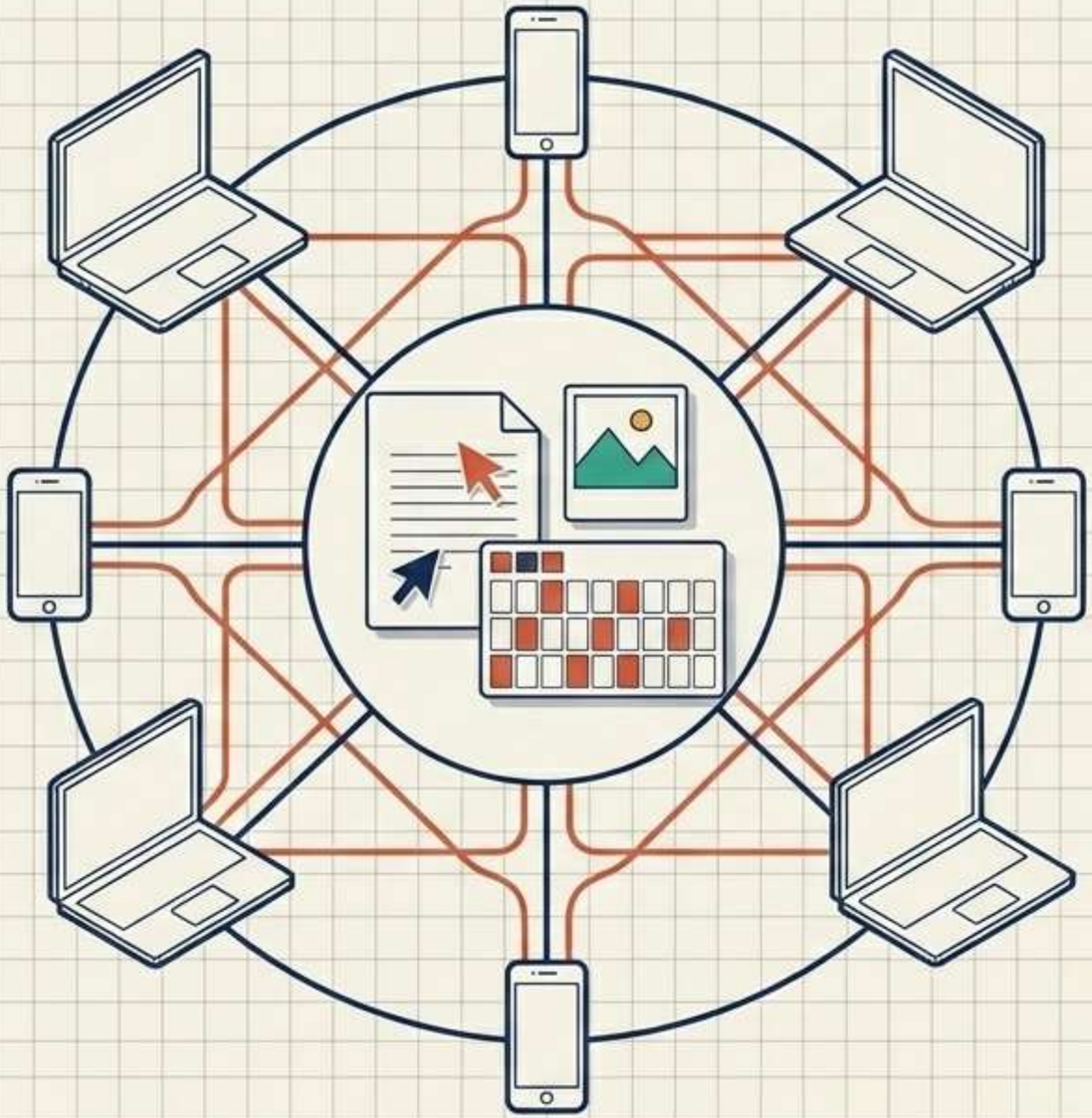
WebRTC Casting

Peer-to-peer camera and screen broadcasting directly to room viewers without server bottlenecks.

Digital Art & 3D

Collaborative spray painting on shared screens (/graffiti/) or manipulating high-fidelity 3D models using mobile gyroscope sensors (/3dviewer/).

Seamless, database-free digital collaboration



Zero-Storage Media

Securely transfer multi-gigabyte files (/files-pro/) by automatically splitting them into chunks, or share photos (/gallery/) that vanish the moment the room closes.

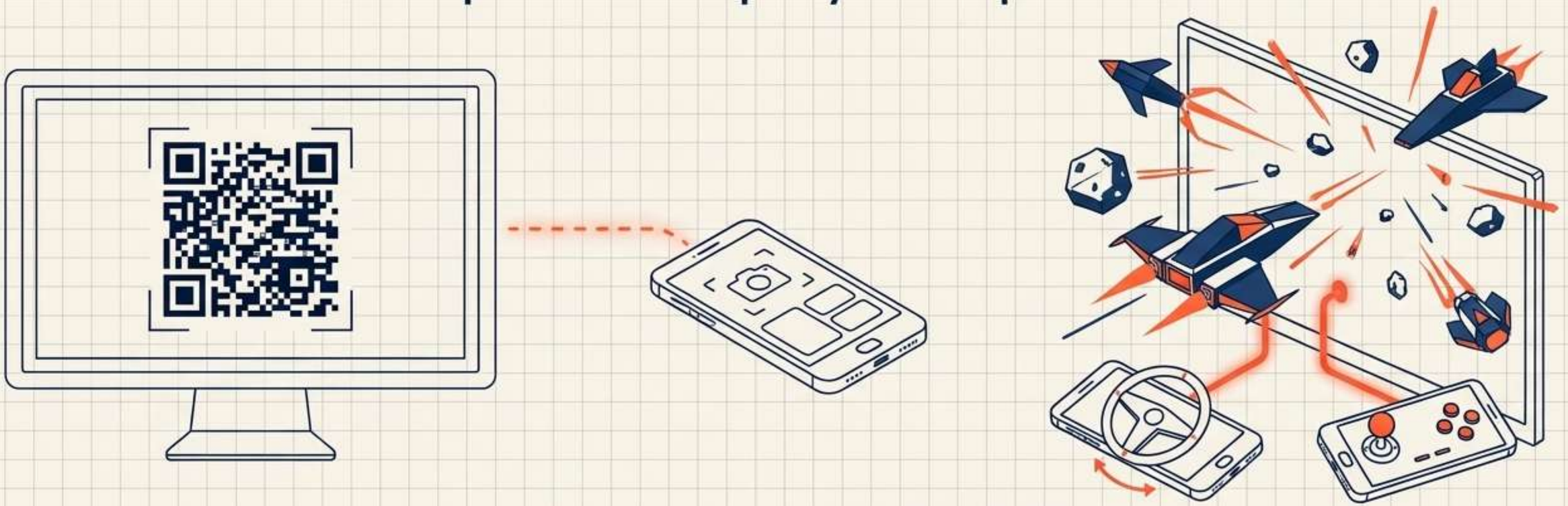
Real-Time Creativity

Shared interactive canvases (/whiteboard/) and synchronised 16-step musical sequencers (/drums/).

Instant Presence

Text editors (/notepad-pro/) with real-time cursor presence and Operational Transformation mechanics, functioning purely peer-to-peer.

Frictionless drop-in multiplayer experiences



The Spacewar Evolution

Demonstrating extreme architectural flexibility, scaling from a basic DOM-based app to a fully hardware-accelerated 3D wave defence game.

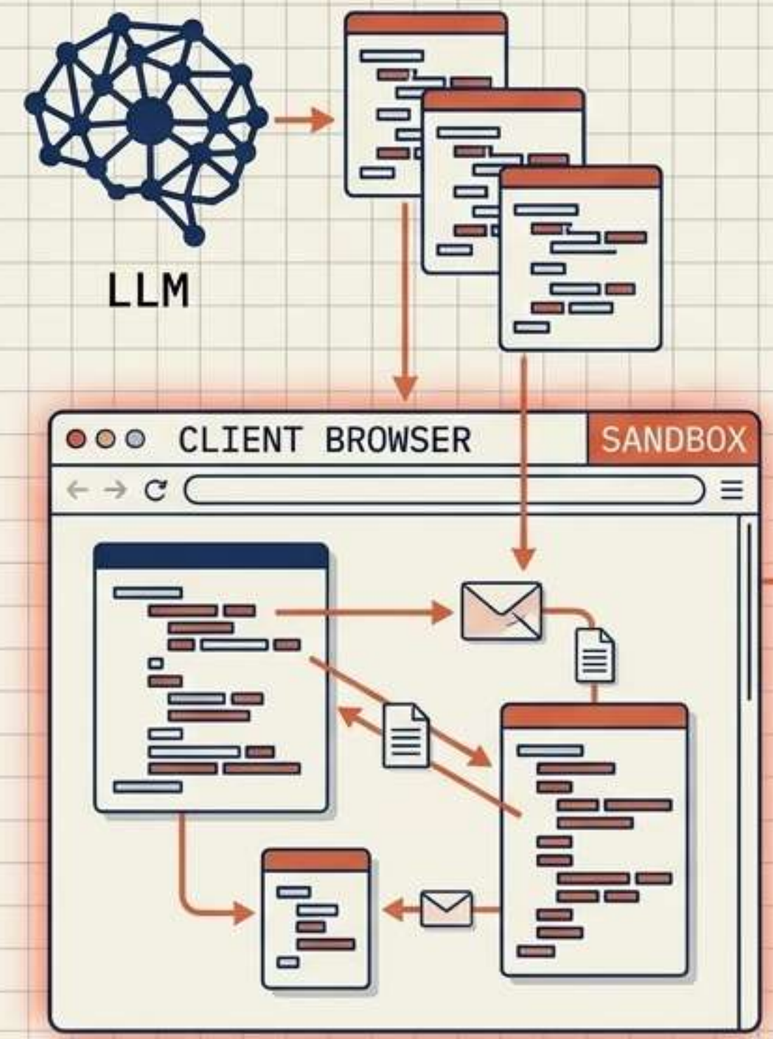
Hardware Sensors

Leveraging mobile physics implementations to turn smartphones into analog steering wheels (**/racer/**) or 6-axis flight yokes (**/flight/**).

Massive Local Arcades

Dozens of users controlling their own snake (**/snake-pro/**) or dropping bombs (**/bomberman/**) in fast-paced procedural labyrinths.

The perfect sandbox for AI-generated code



ISOLATED EXECUTION ENVIRONMENT

SERVER PROXY REMAINS UNTOUCHED

JAVA BACKEND

Absolute Server Safety

The proxy server never parses application data. A broken AI app physically cannot compromise or crash the Java backend.

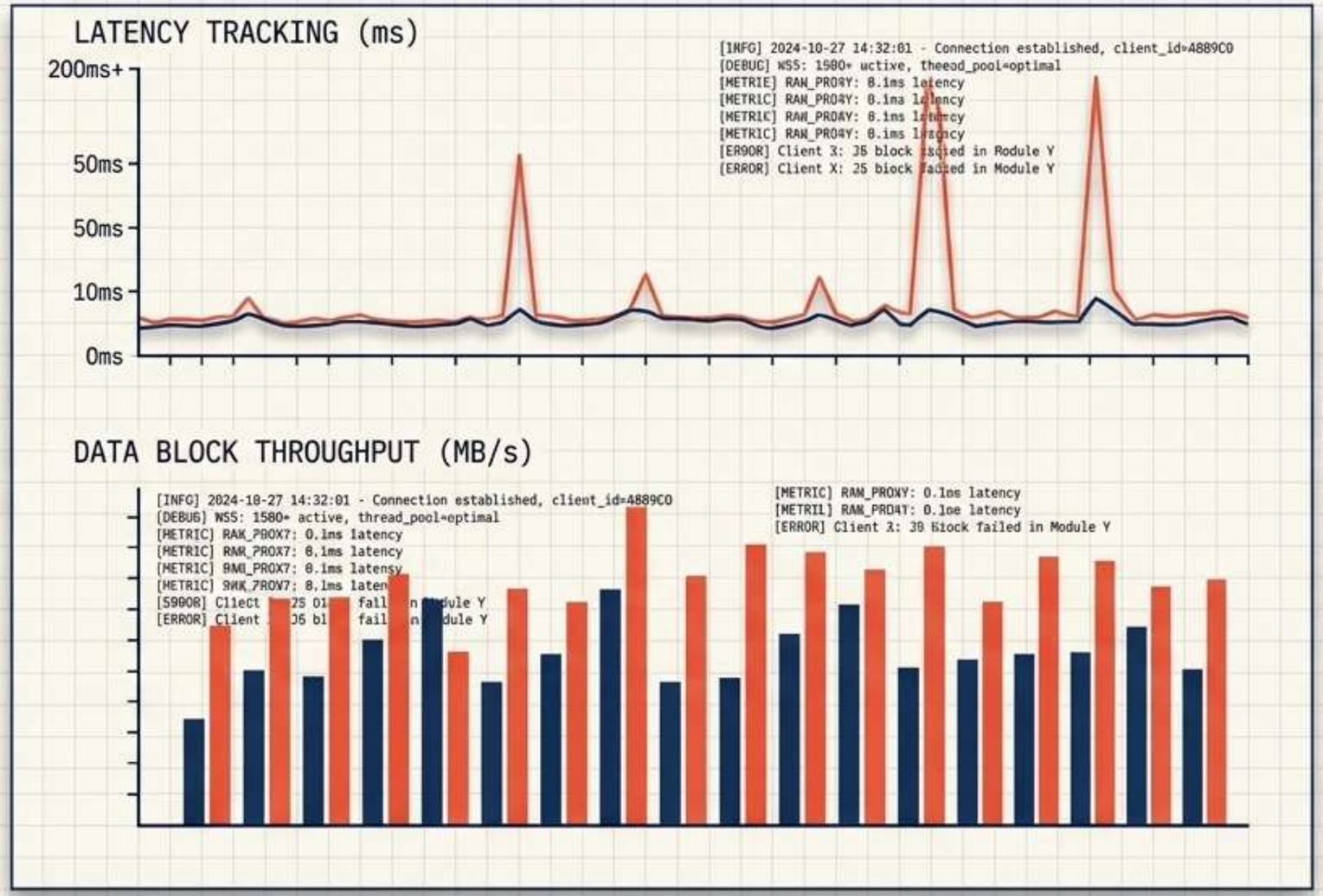
Instant Iteration Loop

Deploying AI code requires zero build steps or server recompilation. Serve static HTML/JS files, and simply reload the browser.

Zero Dependency Hell

Execution errors are isolated completely to the client-side console, allowing for direct stack-trace feedback to the LLM.

High-throughput telemetry and performance



Java Virtual Threads

Custom Java web server maintains thousands of concurrent WebSocket connections with minimal resource overhead.

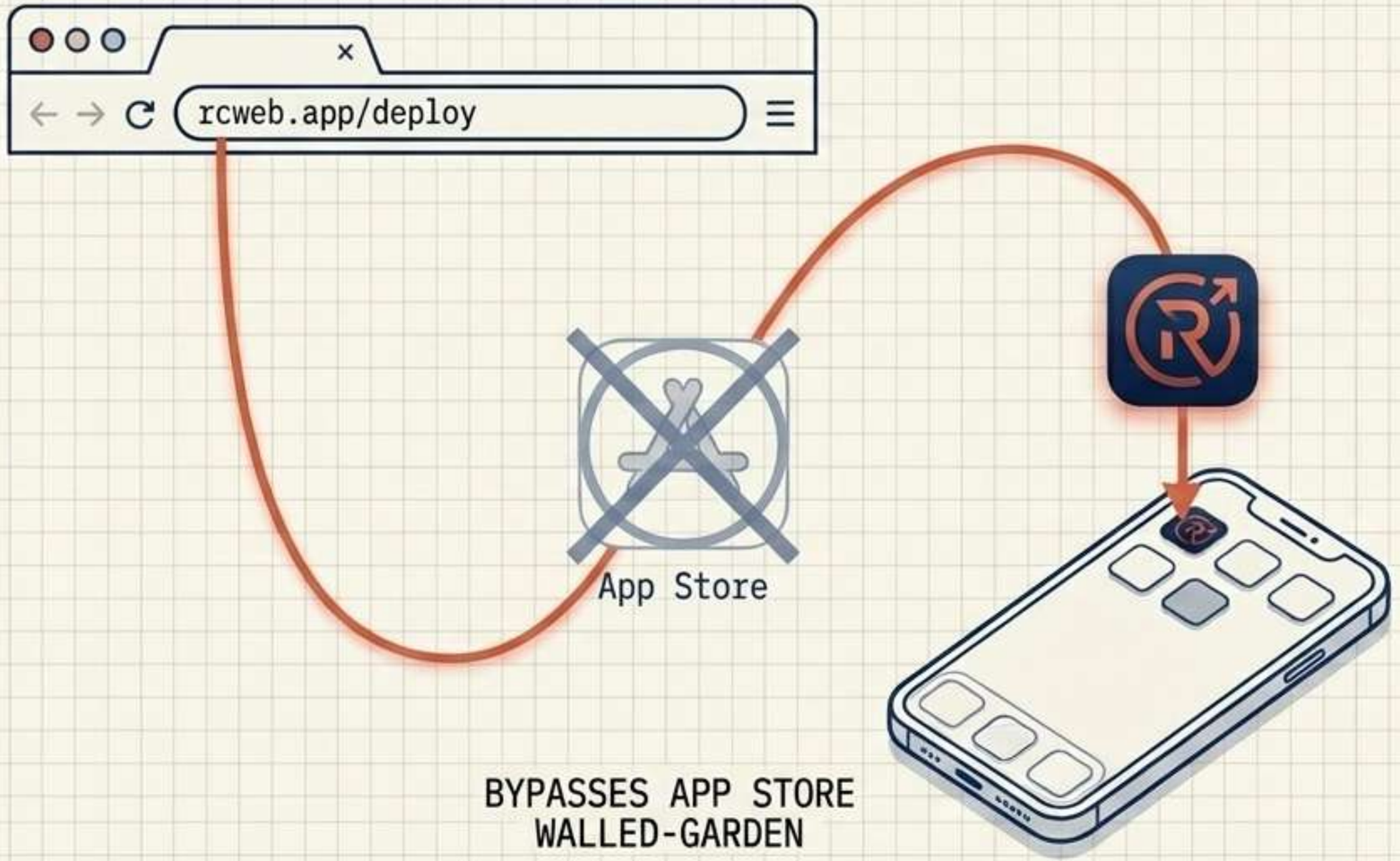
Raw Byte Proxying

Bypasses complex parsing operations entirely, resulting in near-zero latency processing.

Client Telemetry Capabilities

Detailed tracking of latency charts per client, JS block execution success/failure rates, and client environmental data.

Zero-friction deployment via Progressive Web Apps



✓ **INSTANT ACCESSIBILITY**

No app store downloads, approvals, or installations required.

✓ **NATIVE FEEL**

Installable directly to user home screens or start menus.

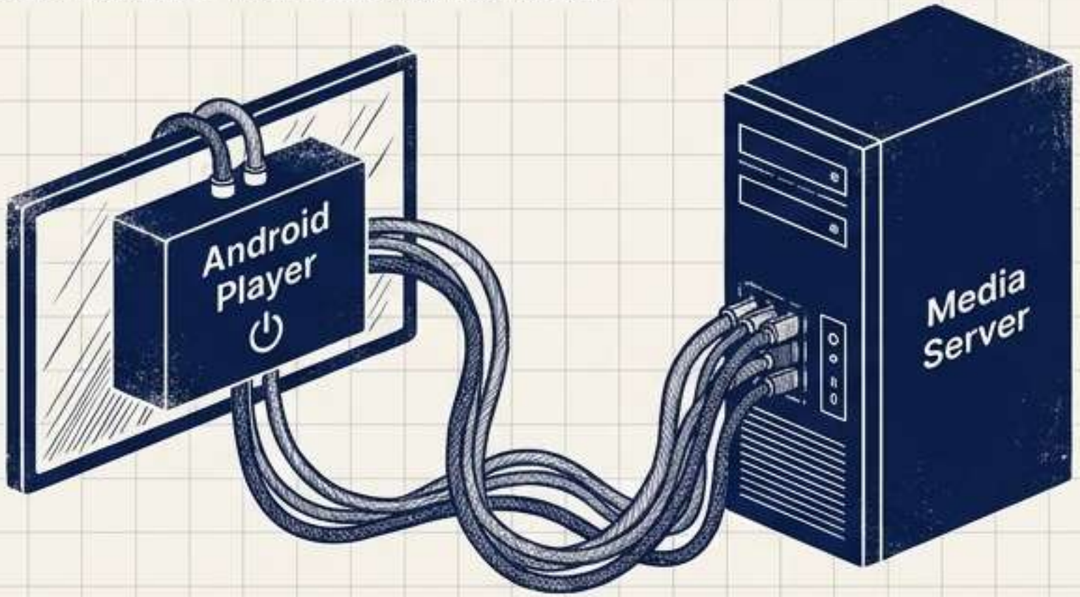
✓ **UNIVERSAL COMPATIBILITY**

Any device with a modern web browser instantly becomes part of the RCWeb ecosystem.

Replacing heavy hardware in enterprise environments

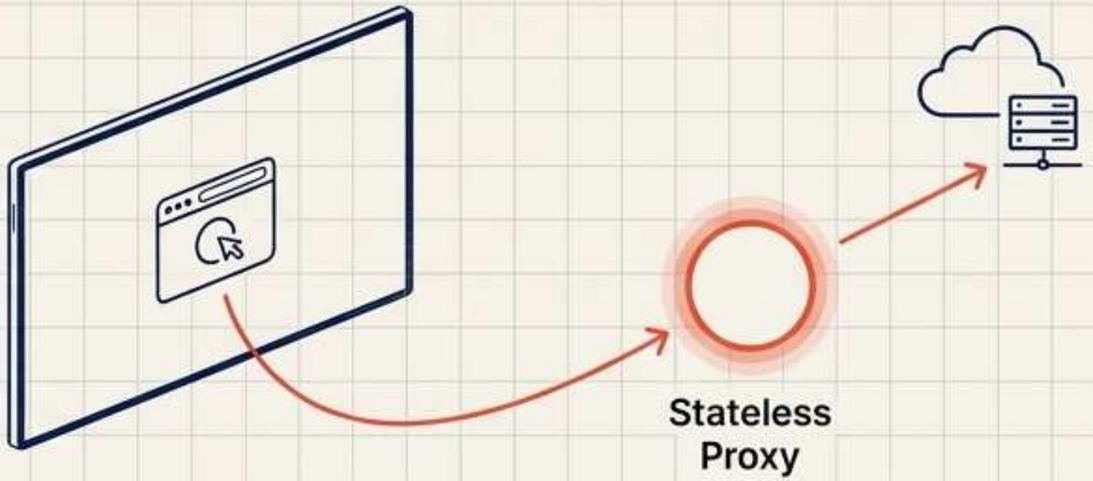
Traditional Digital Signage

Heavy local player boxes, physically tethered screens, and massive multimedia servers.



RCWeb Signage Vision

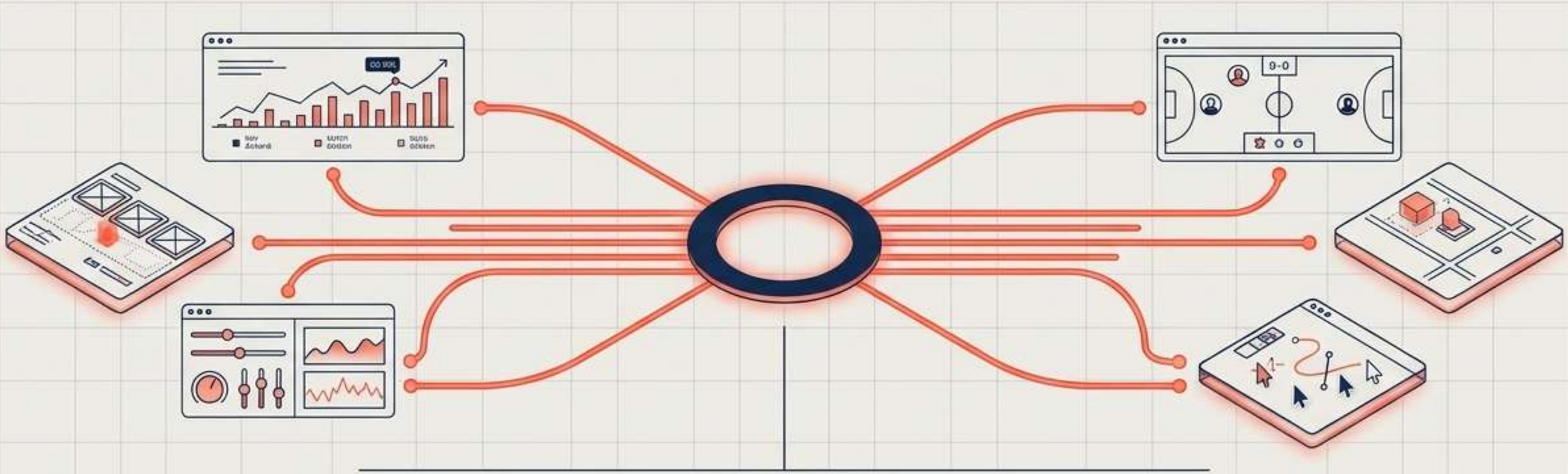
Bare-bones smart displays running simple browsers, orchestrated globally by lightweight, stateless proxies.



Lightweight Digital Signage

Current enterprise signage requires powerful local players and heavy servers to manage multimedia content. RCWeb provides the foundation for a vastly lighter platform, requiring nothing more than a basic web browser and minimal server resources to orchestrate complex, synchronised global displays.

The invisible network waiting for your code



A stateless proxy. Infinite client-side potential.
Build real-time, distributed, multiplayer systems
using only Vanilla JavaScript and the browser.

The invisible network waiting for your code

```
session_status: ready // deploy_code
```