Assetto Corsa — Telemetry & Stats Platform

Personal Project | Full-Stack Real-Time Telemetry System

C++ · Unix IPC · TypeScript · Next.js 14 · PostgreSQL · SSE · React · Real-Time Systems

PROJECT OVERVIEW

Designed and built a production-grade, end-to-end real-time telemetry and race analytics platform for Assetto Corsa dedicated servers. The system integrates a custom C++ telemetry server — communicating over Unix domain sockets — with a Next.js 14 web application that streams live race data to multiple concurrent browser clients. The architecture mirrors the low-latency data pipeline challenges central to iRacing's own telemetry and live timing infrastructure.

RELEVANCE TO IRACING PLATFORM ENGINEERING

This project directly reflects the engineering challenges iRacing solves at scale:

- **IPC & Cross-Layer Integration:** Bridging a compiled C++ data source (AC server plugin) with a web-tier consumer, the same IPC boundary that exists between iRacing's sim binary and its telemetry/data services.
- **Binary Protocol Design & Parsing:** Designing a binary wire protocol (fixed-width C struct layout, LE floats, packed integers, null-terminated strings) and writing a bit-accurate parser,fundamental to iRacing's UDP telemetry API and *ibt* lap file format.
- **Real-Time Fan-Out Architecture:** Building a fan-out pub/sub system to broadcast telemetry packets to N concurrent SSE clients with sub-second latency, equivalent to iRacing's real-time race data distribution.
- **Coordinate Transforms & Track Geometry:** Projecting 3D world coordinates (X/Y/Z floats from the physics engine) onto a 2D PNG track map using track-specific scale, offset, and aspect ratio transforms parsed from INI config files.
- **Telemetry + Persistent Stats Integration:** Integrating live telemetry streams with a persistent PostgreSQL stats layer: per-driver GUIDs, lap histories, rankings, and multi-server session metadata: analogous to iRacing's stat tracking and iRating system.

TECHNICAL IMPLEMENTATION

C++ Telemetry Server & IPC

- Custom C++ server plugin hooks into the Assetto Corsa server process and emits structured binary packets over a Unix domain socket at high frequency.
- Packet layout: 2-byte header (server_id + car_count) followed by 227-byte per-car structs: carID, driver GUID/name, car model (null-terminated, 64-byte fields), 3D world position (3× LE float32), normalised spline position, speed, gear, RPM, lap times, and position rank.
- Socket path: /tmp/ACtelemetry_socket, zero-copy local IPC, no TCP overhead.

TypeScript Bridge Layer (telemetryBridge.ts)

- Singleton TelemetryBridge connects to the Unix socket, reassembles fragmented TCP-style stream data into complete packets via a rolling Buffer, and dispatches to registered subscribers.
- Binary parsing implemented with explicit byte-offset tracking, matching the C struct memory layout precisely, same discipline required when reading iRacing's .ibt telemetry files or iRacing's UDP packet structures.
- Automatic reconnection with exponential backoff; subscriber pub/sub with typed callbacks; SIGTERM/SIGINT cleanup handlers for safe process shutdown.

Server-Sent Events API (Next.js Route Handler)

- SSE endpoint fans out telemetry to concurrent browser clients; each connection subscribes to the bridge and receives per-frame updates streamed as JSON events.
- Driver ranking is overlaid at the SSE layer via a TTL-cached PostgreSQL query (ROW_NUMBER OVER ORDER BY user_rank DESC), avoiding a DB hit per-packet while staying fresh.
- Heartbeat keepalive every 30s; per-connection abort signal cleanup to prevent subscriber leaks; nginx buffering disabled via X-Accel-Buffering header.

Track Map & Coordinate System

- Parses AC's map.ini format (WIDTH, HEIGHT, SCALE_FACTOR, X_OFFSET, Z_OFFSET, DRAWING_SIZE) to reconstruct the same coordinate transform AC uses internally.
- worldToMapCoords() applies aspect-ratio correction, offset translation, scale division, and percentage normalisation to project physics-engine 3D positions onto 2D SVG space — reverse-engineered from AC's rendering pipeline.
- Falls back to normalised spline position for tracks without a map.ini, ensuring graceful degradation.

Database & Stats Layer

- PostgreSQL connection pool (max 20 connections) with singleton pattern; typed query helpers; graceful shutdown with pool.end().
- Schema tracks driver GUIDs, lap times, user rankings, session types (Practice/Qualifying/Race), track conditions (ambient/road temp), and multi-server session metadata.
- Event registration and results system with per-driver historical data — foundation for a competitive rating system analogous to iRating/Safety Rating.

STACK & TECHNOLOGIES

Backend: C++, Unix sockets, Next.js 14 (Node.js runtime), TypeScript, PostgreSQL (pg pool)

Frontend: React (Server + Client Components), SSE, SVG, Tailwind CSS

Protocol / Data: Custom binary protocol over Unix IPC, JSON-over-SSE, INI config parsing, PostgreSQL SQL

Infrastructure: Linux, nginx (reverse proxy, SSE passthrough), multi-server session management