System Architecture Specification

Cross-Exchange High-Frequency Arbitrage (HFA) Engine

---

1. Executive Summary

This document outlines the architectural framework of an automated high-frequency trading (HFT) system designed for cross-venue liquidity capture. The system utilizes low-latency WebSocket protocols and specialized blockchain distribution networks (BDN) to aggregate real-time order book data, decoded mempool intent, and finalized block transactions across Centralized Exchanges (CEXs) and Decentralized Exchanges (DEXs).

2. Technical Architecture Diagram

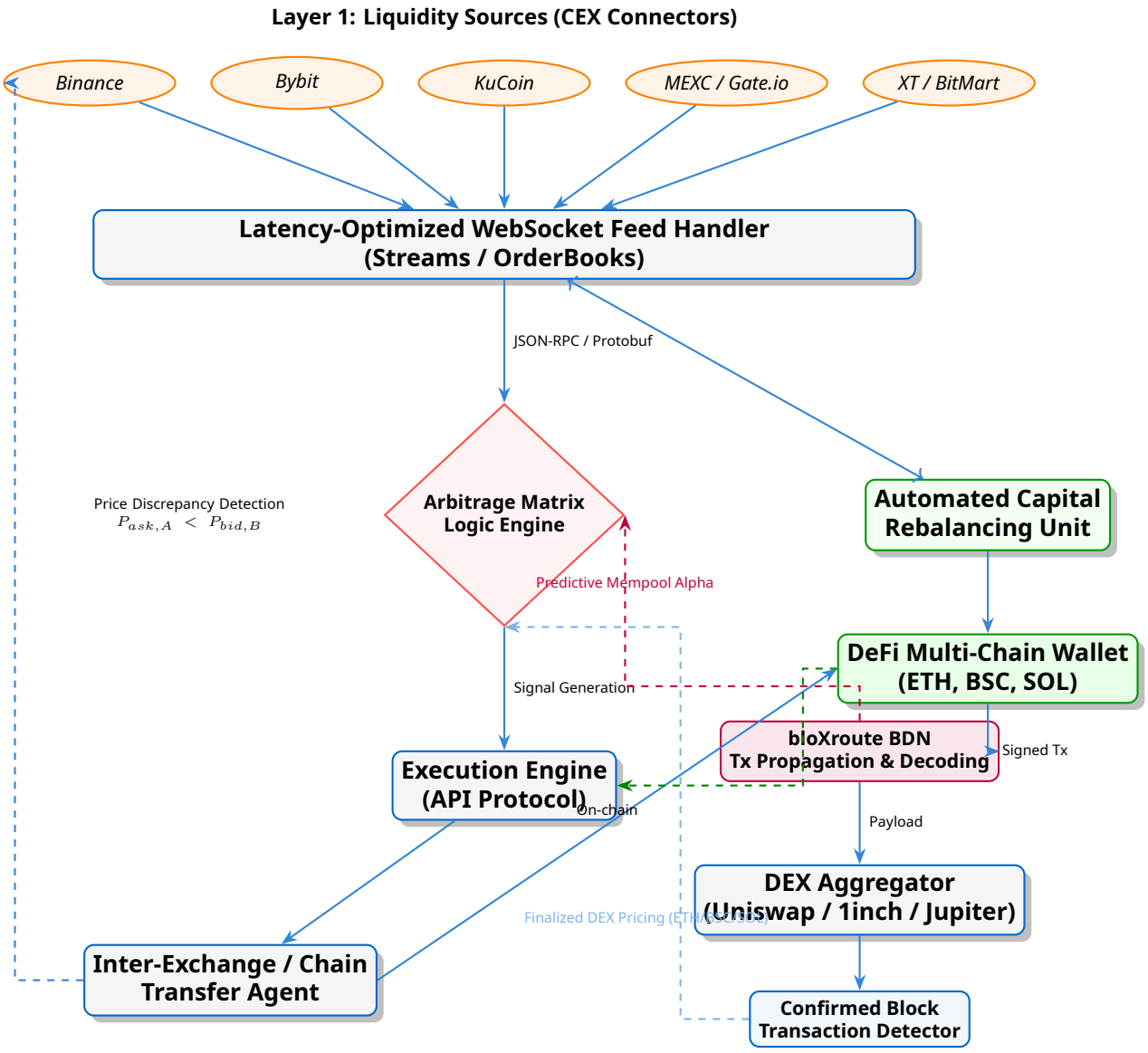

Figure 1: Full-stack HFA System Flow with Dual-Layer On-Chain Feedback (Mempool + Block Detection).

3. Functional Components

3.1 Multi-Venue WebSocket Ingestion

The system establishes persistent TCP connections to high-liquidity venues. The ingestion layer processes L2 order book updates with a median latency of < 5ms.

3.2 The Arbitrage Logic Matrix (Quad-Feed Input)

The engine maintains a multi-dimensional state matrix influenced by four distinct data streams:

1. **CEX Feeds:** Real-time order book snapshots via WebSockets.
2. **Mempool Alpha (bloXroute):** Pre-block transaction decoding to predict DEX price shifts.
3. **Confirmed Block Detector:** A specialized listener that parses finalized block data across ETH, BSC, and SOL. It identifies third-party swaps that have landed on-chain, providing absolute ground-truth for DEX pricing.
4. **Internal Rebalancing State:** Monitoring of internal capital movement across venues.

3.3 bloXroute BDN & Mempool Decoding

Utilizes bloXroute's Cloud-API to stream the global mempool.

- **Real-Time Decoding:** The engine decodes ABI data for large swaps on Uniswap, 1inch, and Jupiter to identify front-running or follow-on arbitrage opportunities.
- **Propagation:** Signed transactions are broadcast via the bloXroute BDN to ensure sub-millisecond propagation to global validator sets.

3.4 Multi-Chain Execution (ETH, BSC, SOL)

The system monitors and executes trades across three major ecosystems. Gas estimation is performed dynamically per block, ensuring that `Gas_Transfer` and `Gas_Swap` costs are factored into the Δ_{Profit} calculation in real-time.

4. Operational Infrastructure

Nodes are optimally placed geographically for minimal latency. The logic engine is built using Node.js using master/worker clusters for high-concurrency event handling, utilizing optimized `ethers.js` providers for low-overhead blockchain interactions.