Project 1: Multiplayer Tetris

Objective

Design and implement a decentralized multiplayer Tetris game where players interact directly via peer-to-peer (P2P) connections. The system will use gossip protocols for state synchronization, Conflict-Free Replicated Data Types (CRDTs) for mergeable game states, and digital signatures to prevent malicious behavior. This project will demonstrate practical applications of distributed consensus, fault tolerance, and secure communication.

Motivation

Traditional multiplayer Tetris games rely on centralized servers, introducing latency, scalability limits, and single points of failure. This project addresses these issues by:

1. Eliminating Central Servers: Players connect directly, reducing dependency on costly infrastructure.
2. Ensuring Consistency: CRDTs and logical clocks synchronize game states across partitions.
3. Preventing Cheating: Cryptographic signatures validate all moves.

Key Features

1. State Sync via UDP
    a. Peers exchange game state deltas via UDP (lower latency than TCP).
2. CRDT Game Board
    a. Merge piece placements and line clears automatically.
3. Byzantine-Resistant Moves
    a. Cryptographic signatures verify each player's actions.
4. Logical Clock Synchronization
    a. Use logical clock timestamps to order player actions.
5. Dynamic Game Room Creation
    a. Players will be able to create private rooms and join these rooms using a code.

Additional Features (if time allows)

1. Blockchain Leaderboard
    a. Store high scores on secure ledgers.

Implementation Plan

Week	Milestone	Deliverables
1	Core Gameplay + P2P Networking	Single-player Tetris with WebRTC

2	CRDT State Sync + Gossip Protocol	Multiplayer board merging
3	Player move security + NAT Traversal (for direct communication)	Signature validation, STUN setup
4	Chaos Testing + Demo	Engineering notebook, multiplayer demo

Evaluation Metrics

1. Move latency
2. State convergence
3. Malicious move detection + rejection
4. NAT success rate

Impact + Deliverables

This project will demonstrate how P2P architectures can replace centralized gaming infrastructure. The result of this project will be a playable demo and engineering notebook documenting the steps taken to build the system and challenges encountered.

Project 2: Decentralized P2P AI Network

The AIDO (AI-Driven Decentralized Organization) system presents an innovative approach to decentralized decision-making using AI agents. Our goal with this project is to analyze its implementation and distributed systems architecture and propose enhancements.

System Analyzed: AI-Driven Decentralized Organization (AIDO)
Repo: https://github.com/ruvnet/aido
Focus Areas: Consensus mechanisms, fault tolerance, and task allocation in AI-driven distributed systems

Key Components Analysis

Component	Current Implementation (AIDO)	Distributed Systems Concepts
Consensus	Average scoring system (7+ accepts)	Similar to eventual consistency; lacks Byzantine fault tolerance
Communication	Supabase RPC calls between agents	Follows client-server model; could adopt peer-to-peer patterns
Task Allocation	Single LLM decision maker	Centralized coordinator pattern; could implement decentralized auction system
Failure Handling	Basic retries in function code	No apparent quorum systems or redundancy mechanisms

Proposed Improvements

1. Enhanced Consensus Mechanism

Current Limitation: Simple average scoring is vulnerable to Sybil attacks and collusion
Proposed Solution:

- Implement a reputation-weighted voting system inspired by Practical Byzantine Fault Tolerance (PBFT)
- Add proof-of-work-style computational puzzles for critical decisions
- Introduce dynamic quorum sizing based on proposal complexity

2. Distributed Task Allocation

Current Limitation: Centralized LLM makes allocation decisions
Proposed Solution:

- Implement a decentralized auction system using conflict-free replicated data types (CRDTs)

- Add capability-based routing using agent skill matrices
- Introduce fallback agents using eventually consistent replica sets

3. Failure Recovery System

Current Limitation: No apparent node failure handling
Proposed Solution:

- Implement heartbeat monitoring with exponential backoff.
- Add replica agent pools using the Raft consensus algorithm
- Introduce decision rollback capabilities using operational transforms

Evaluation Metrics

1. Fault tolerance ratio under network partitions
2. Consensus convergence time with varying node counts
3. Task allocation latency distribution
4. Recovery time from agent failures

This analysis/implementation would encompass a fast-growing field and demonstrate how AIDO's AI-driven approach with fundamental distributed systems principles could improve through grounded classical distributed algorithms. The project would implement and evaluate these enhancements using the existing codebase as a foundation. This is a rough idea of the direction; we may check other repositories and projects in the future.